UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

ESEIAAT

Grado en Ingeniería en Vehículos Aeroespaciales

ESTUDIO DE UN SISTEMA DE AUTOMATIZACIÓN ENFOCADO A LA SOSTENIBILIDAD ENERGÉTICA

Memoria

Estudiante: Nil Jarque García

Director: Miquel Delgado Prieto

Codirector: Ángel Fernández Sobrino

Fecha de entrega: Enero 2020

Agradecimientos

A mi familia y mis amigos,
Por su apoyo y soporte constante.

A mis directores, Miguel Delgado Prieto y Ángel Fernández Sobrino,
Por guiarme y ayudarme en la realización del proyecto.

A toda la comunidad involucrada en software de código abierto,
Sin la cual este proyecto no hubiese sido posible.

Resumen

Este estudio considera el diseño y desarrollo de un sistema IoT (Internet de las cosas, del inglés *Internet of Things*), escalable, robusto y a prueba de fallos, para monitorizar varias variables físicas relacionadas con la monitorización de un sistema HVAC (Ventilación, calefacción y aire acondicionado, del inglés *Heating, Ventilating and Air Conditioning*), como ejemplo la concentración de CO_2 y la corriente electica consumida.

Los datos recogidos tienen el objetivo de ser utilizados para valorar el estado de confort de las personas que se encuentran en el interior de un edificio según la calidad del aire de este, así como valorar el perfil de consumo del sistema conectado a la distribución eléctrica.

Para conseguir esto, en este trabajo se considera una programación a prueba de fallos del dispositivo digital, la fiabilidad de los protocolos de comunicación elegidos, una arquitectura adecuada para permitir la ampliación del sistema y el procesado de los datos.

En este trabajo, el sistema de comunicación debe permitir que la monitorización de los datos sea accesible por internet desde cualquier lugar. En este sentido, el sistema se decide que esté formado por un microcontrolador de bajo coste con capacidades WiFi, que permite el pre-procesado local y la comunicación mediante protocolo HTTP (i.e. *Hyper Text Transfer Protocol*), a una base de datos. Finalmente, se considera una plataforma de visualización para la presentación y monitorización de los datos. Hay que considerar que la plataforma de visualización hace peticiones a la base de datos también mediante protocolo HTTP, y muestra gráficamente los resultados permitiendo el post-procesado de datos, así como la creación de alarmas y eventos.

La red de sensores inalámbricos, WSN (i.e. *Wireless Sensor Network*), utilizada se compone de un sensor de CO_2 que mide la concentración de este gas en el aire, así como de una pinza amperimétrica que mide de forma no intrusiva la corriente alterna usada por el sistema HVAC u otra eventual carga.

Así, por un lado, el sensor de CO_2 permitirá conocer indirectamente la ocupación de la habitación y calidad del aire, IAQ (i.e. *indoor air quality*). Por otro lado, la pinza

amperimétrica permite estimar el consumo energético, así como evaluar la calidad de la red eléctrica.

Los datos obtenidos permiten crear alarmas que se activen a través de eventos (e.g. consumo muy alto, contenido de CO_2 muy alto, etc.), y visualizar un historial evolutivo de los valores. Con estos datos es posible realizar distintos cálculos de uso y simulaciones de tendencias, lo que permite tomar decisiones en cuanto a la administración del edificio.

Esta arquitectura, basada en hardware de bajo coste y programas de código abierto, permite la adición de nuevos sensores, es fácilmente escalable y puede servir de base para multitud de funcionalidades, por lo que representa una solución transversal con potencial de aplicación a otros sectores y aplicaciones.

Abstract

This study proposes a scalable, robust and fail-safe IoT (Internet of Things) communication system to monitor various physical variables, related with the monitoring of an HVAC (Heating, Ventilating and Air Conditioning) system, i.e. the concentration of CO_2 and the consumed power.

The collected data can be used to assess the comfort of people inside a building according to the indoor air quality, as well as monitoring characteristics of the electrical grid. Finally, a post-processing of the data is included.

This study aims to achieve this by programming a device in a fault tolerant way, using a suitable architecture to enable system expansion and taking into account the reliability of the chosen protocols.

The communication system enables the monitoring of data to be accessible through internet from anywhere. The system is mainly composed of a low cost microcontroller with native WiFi capabilities, which processes the data and enables communication with a database using HTTP (Hyper Text Transfer Protocol). Finally, a data visualization platform is considered for data monitoring. This platform makes requests to the database using HTTP and graphically displays the results enabling some post processing, various visualizations and the creation of alarms and events.

The used wireless sensor network (WSN) is composed of a CO_2 sensor that measures the concentration of this gas as well as a current transformer that can measure the current used by the HVAC system or any wire in a non-intrusive way.

The CO_2 sensor will allow knowing indirectly the occupation of the room and quality of the air (IAQ, indoor air quality). The current transformer makes it possible to estimate energy consumption, as well as to evaluate the quality of the electrical network.

The obtained data can trigger alarms configured by means of events like a very high current draw, high concentration of CO_2 etc. As the data is stored in the database, the historic values can be shown, and it is possible to post-process them to obtain tendencies and various important usage values. This can help make decisions regarding the administration of the building.

The obtained data can trigger alarms configured by means of events like a very high current draw, high concentration of CO_2 etc. As the data is stored in the database, the historic values can be shown, and it is possible to post-process them to obtain tendencies and various important usage values. This can help make decisions regarding the administration of the building.

The system architecture is based on low cost hardware and open source programs, with the objective of making the addition of new sensors easy as well as making the system scalable and flexible, serving as flexible a basis for multitude of IoT projects in different fields.

Declaración de honor

Declaro que,

el trabajo en esta Tesis de Grado es completamente mi propio trabajo,

ninguna parte de esta Tesis de Grado se ha tomado del trabajo de otras personas sin darles mención

todas las referencias han sido claramente citadas.

Entiendo que la infracción de esta declaración me deja sujeto a las acciones disciplinarias previstas por la Universitat Politècnica de Catalunya - BarcelonaTECH.

Nil Jarque Enero 2020

Título de la tesis:

Estudio de un sistema de automatización enfocado a la sostenibilidad energética

Contenido

Lista de abreviaturas

ADC	Conversor analógico digital *Analog to Digital Converter*
API	Interfaz de programación de la aplicación *Application Programming Interface*
ARP	Protocolo de resolución de direcciones *Address Resolution Protocol*
ASCII	Código Estándar Estadounidense para el Intercambio de la Información *American Standard Code for Information Interchange*
BIM	Modelado de información de construcción *Building information modeling)*
CLI	Interfaz por línea de comandos *Command Line Interface*
DDE	Intercambio dinámico de datos *Dynamic Data Exchange*
EPI	Equipo de protección individual
FFT	Transformada de Fourier rápida *Fast Fourier Transform*
GSMA	Asociación sistema global para las comunicaciones móviles *Global System for Mobile Communications Association*
GUI	Interfaz gráfica de usuario *Graphical User Interface*
HTTP	Protocolo de transferencia de hipertexto *Hypertext Transfer Protocol*
HTTPS	HTTP Seguro *Secure HTTP*

HVAC	Ventilación, calefacción y aire acondicionado *Heating, Ventilating and Air Conditioning*
IAQ	Calidad del aire en interiores *Interior Air Quality*
IDE	Entorno de desarrollo integrado *Integrated Development Environment*
InfluxQL	Lenguaje estructurado Influx *Influx Query Language*
IoT	Internet de las cosas *Internet of Things*
IP	Protocolo de internet *Internet Protocol*
ISO	Organización Internacional de Normalización *International Organization for Standardization*
LAN	Red de área local *Local Area Network*
MAC	Control de acceso al medio *Media Access Control*
MBPC	Control predictivo por modelo *Model-Based Predictive Control*
NoSQL	No SQL
OSI	Modelo de interconexión de sistemas abiertos *Open System Interconnection*
RFC 791	Petición de comentarios *Request For Comments*
RFID	Identificación por radiofrecuencia *Radio Frequency Identification*
RMS	Valor cuadrático medio

	Root Mean Square
SBC	Ordenador de una sola placa *Single Board Computer*
SDK	Kit de desarrollo de software *Sofware Development Kit*
SPI	Interfaz periférica serial *Serial Peripheral Interface*
SQL	Lenguaje de consulta estructurada *Structured Query Language*
TCP	Protocolo de control de transmisión *Transmission Control Protocol*
TLS	Seguridad en la capa de transporte *Transport Layer Security*
TRS	Punta-Anillo-Cuerpo *Tip-Ring-Sleeve*
TSDB	Base de datos temporal *Time Series Database*
UDP	Protocolo de datagramas de usuario *User Datagram Protocol*
URL	Localizador de recursos uniforme *Uniform Resource Locator*
WDT	Temporizadores perrito-guardián *Watch Dog Timers*
WSN	Red de sensores inalámbricos *Wireless Sensor Network*
WWW	Red informática mundial *World Wide Web*

Índice de figuras

Índice de tablas

1 Prefacio

La definición de IoT ha ido evolucionando con el tiempo debido al avance de las tecnologías de análisis de datos, sensores y dispositivos integrados. En general, el IoT se define como un sistema de dispositivos con un identificador único y la capacidad de transferir datos a través de una red sin intervención humana.

Los dispositivos IoT engloban desde *smartphones*, *tablets*, *smartwatches* y otros *wearables* a simples dispositivos como juguetes, actuadores, lectores de RFID (Identificación por radiofrecuencia, del inglés *Radio Frequency Identification*) o dispositivos integrados como microcontroladores que utilizan la red para subir datos a la nube.

El internet de las cosas tiene un gran potencial y sus aplicaciones están creciendo, usando nuevos dispositivos que cada vez son más flexibles, eficientes y más rentables económicamente. Todos los dispositivos que se puedan beneficiar de una conexión, se conectarán entre ellos de la mejor manera posible, revolucionando la forma de interactuar con el mundo físico y con el resto de personas.

De esto se desprende que cada vez se idean más soluciones usando los distintos dispositivos existentes, por ejemplo: en sistemas inteligentes, agricultura, transporte, salud, industria, domótica o monitorización. Según la GSMA (i.e. *Global System for Mobile Communications Association*), en 2018 el número de conexiones de dispositivos IoT superaba los 9.000 millones, mientras que se espera que en 2025 se sobrepasen los 25.000 millones. (Mark Page ; Maria Molina, 2019)

Por lo tanto, es vital que cada dispositivo IoT se conecte de una manera eficaz y eficiente, adaptada a su función.

Consecuentemente, en este estudio, se hará especial énfasis en las características de los protocolos de comunicación usados, así como en la justificación de las soluciones de software escogidas y, en general, de toda la estructura de comunicación que se diseña y se desarrolla.

2 Introducción

2.1 Objetivo

Estudio de las etapas de diseño, desarrollo y validación de un sistema IoT para la monitorización de perfiles de consumo y el diagnóstico sobre la distribución eléctrica asociada a los sistemas HVAC (i.e sistemas de ventilación, calefacción y aire acondicionado, del *inglés Heating, Ventilating and Air Conditioning*), e iluminación, así como características relacionadas con la calidad del aire aplicadas a edificios del sector terciario.

2.2 Alcance

El alcance del estudio considera el diseño y el desarrollo de un prototipo de dispositivo integrando instrumentación y electrónica disponible en el mercado.

El estudio considerará especialmente el despliegue y el diseño de la arquitectura de comunicaciones necesaria para el registro, procesado y monitorización remota de la información.

También se delimita el entorno a estudiar, considerando no un edificio completo, sino unos espacios específicos.

2.3 Requerimientos

El microcontrolador tendrá que comunicarse mediante WiFi con un ordenador central, y finalmente la información recogida por los sensores ha de mostrarse a través de internet, siendo necesario que se presente de una manera provechosa y disponible a cualquier dispositivo con acceso a internet.

El sensor de corriente utilizado será el SCT-013 debido a que sus características son las apropiadas para el proyecto y por su disponibilidad en el laboratorio desde el inicio del trabajo.

2.4 Estructura

El presente documento se estructurará alrededor de las tareas necesarias para llevar a cabo los objetivos planteados, a continuación, se muestran dichas tareas.

2.4.1 Estructura de descomposición de trabajo

A continuación, se presenta la estructura de descomposición de trabajo con las tareas generales a llevar a cabo (Figura 1), también conocida en inglés como Work Breakdown Structure (WBS).

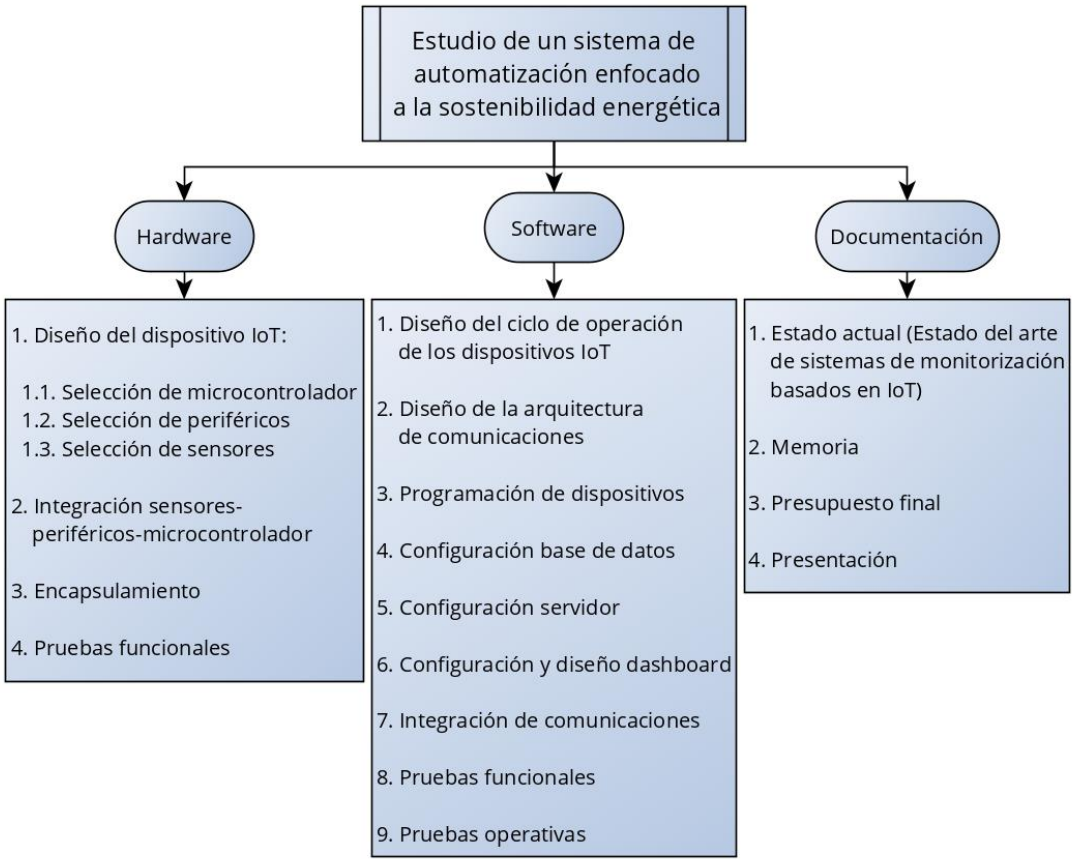

Figura 1: Estructura de descomposición de trabajo

2.4.2 Estructuración del documento

El resto de este documento se organizará de la siguiente manera: el capítulo 3 Estado del arte, explora los diferentes productos comerciales así como artículos científicos relacionados con la aplicación y la tecnología usada. El capítulo 4 Comunicaciones y protocolos, presenta conceptos necesarios para la elaboración del proyecto. En el capítulo 5 Diseño del sistema, se justifica la elección del hardware y software utilizado, así como se presenta el diseño de la arquitectura del sistema. En el capítulo 6 Elaboración del prototipo, se introduce el prototipo y una explicación de los pasos necesarios para su desarrollo. En el capítulo 7 Pruebas y resultados se describen las pruebas realizadas y se muestran los resultados. Finalmente, tras el estudio medioambiental (Impacto medioambiental), y el resumen del presupuesto (Resumen del presupuesto), se dan las conclusiones y se hace una breve discusión de los resultados así como un comentario sobre futuras líneas de trabajo.

3 Estado del arte

En este capítulo se pretende realizar una revisión de los desarrollos hechos hasta ahora dentro del campo de la monitorización energética con dispositivos IoT. El objetivo es establecer un contexto del trabajo ya realizado en esta área: examinar las diferentes ideas existentes con sus puntos fuertes y débiles y así obtener una idea de como se ha abordado este problema en los últimos años.

Para analizar las soluciones actuales, estas se han dividido en dos subgrupos: soluciones comerciales existentes en el mercado actual y trabajos relacionados de investigación.

3.1 Soluciones comerciales

Estas soluciones se caracterizan por estar dirigidas, normalmente, a un público general, siendo de fácil instalación y con una interfaz de usuario simple y estéticamente atractiva, añadiendo más o menos funcionalidades dependiendo del modelo y el precio. Salvo algunas excepciones, la mayoría son sistemas cerrados y patentados, que no permiten su modificación ni la expansión de su usabilidad.

Estos sistemas están enfocados en ofrecer un seguimiento del consumo energético para poder visualizarlo gráficamente, saber donde se encuentran los picos de uso y el gasto que esto produce en particulares. Este no es completamente el objetivo del presente trabajo, pero la mayoría de soluciones comerciales, listadas a continuación, están centradas en la monitorización del consumo eléctrico.

En primer lugar, las propias empresas de distribución eléctrica permiten observar el consumo en función del tiempo, siendo la mayor ventaja para el consumidor que es un servicio gratuito. Las principales limitaciones son que solo se puede ver el consumo del mes anterior y que este aparece con una resolución máxima de una hora.

En segundo lugar, existen algunas alternativas open-source, como Open Energy Monitor (openenergymonitor.org), e IoTaWatt (iotawatt.com).

Open Energy Monitor es un proyecto open-source de monitorización del consumo de varios circuitos eléctricos, emplea una arquitectura modular escalable con diferentes dispositivos periféricos (emonTX, emonPi) que se conectan a uno central (emonBase o emonPi). En este central, se lleva a cabo el procesado de datos, visualización de los mismos y generación de alertas usando su propio software (emoncms). El hardware está basado en Raspberry Pi y Arduino. Su principal desventaja es el precio de compra, de unos 160€ actualmente.

IoTaWatt es otra alternativa open-source y open-hardware basada en el microcontrolador ESP8266. También desarrollado junto con Open Energy Monitor, IoTaWatt es una alternativa a emonTX, con ventajas y desventajas respecto a este: si bien consume más, acepta más sensores para monitorizar más circuitos que emonTX y permite guardar los datos en una tarjeta SD, por ejemplo.

Finalmente, también existen multitud de alternativas de productos comerciales, de las que podemos destacar las siguientes:

Efergy tiene todo un rango de soluciones disponibles en varios precios: dirigidas a medir el consumo de solo un enchufe, o de toda la instalación eléctrica de una vivienda, con un monitor para mostrar la información o mostrándola solo remotamente. También permite integrar varios de sus dispositivos para obtener así aún más información específica, por ejemplo, para distinguir entre los electrodomésticos conectados.

Wibeee (https://wibeee.com/) es otro medidor muy compacto que puede medir varios circuitos eléctricos conectándose al cuadro eléctrico. Estas mediciones son presentadas en una plataforma en internet en la cual se muestra el consumo en tiempo real. Wibeee cuenta con funciones avanzadas como el auto-reconocimiento de los aparatos eléctricos conectados mediante un algoritmo que logra identificarlos observando el cambio en el consumo de potencia eléctrica (más inductiva, capacitiva, etc.).

Netbiter (www.netbiter.com) es otro dispositivo que permite exponer variables recogidas por sensores en distintos *dashboards,* aunque este está más enfocado a la monitorización de entornos industriales.

También existen otros medidores de consumo para enchufes más simples, de marcas distintas, como, por ejemplo, Owl, Wattio, o alguno de los productos de

Efergy. Estas marcas ofrecen productos simples pero útiles, siendo su principal desventaja el precio (50-80€), que hace que el ahorro que se puede conseguir obteniendo una monitorización personalizada quede desplazado por esta inversión inicial, tal como ocurre también con los productos más sofisticados que se han visto anteriormente.

Ninguna de estas opciones puede funcionar como base para montar otro tipo de sensores utilizando su esquema de comunicaciones, aunque las variantes open-source como Open Energy Monitor o IoTaWatt sí permiten subir los datos recolectados a varias bases de datos para su posterior post-proceso o almacenaje.

3.2 Trabajos relacionados de investigación

La mayoría de trabajos relacionados son soluciones para problemas particulares, ideaciones de nuevos tratamientos para la información, pruebas de concepto o actualizaciones/continuación de trabajos anteriores. Por esto, se ha decidido estudiar trabajos publicados en los últimos cinco años, concretamente artículos publicados en revistas científicas o actas de conferencias.

Una buena parte de la investigación se centra en estudiar sistemas HVAC de edificios y como disminuir su consumo, al ser estos sistemas de los más intensivos energéticamente. Así, por ejemplo, en (Ruano, Silva, Duarte, & Ferreira, 2018) se describe el diseño y validación en un edificio de una plataforma IoT basada en una técnica de control inteligente de sistemas HVAC llamada MBPC (Control predictivo por modelo, del inglés *Model-Based Predictive Control*). Las comunicaciones se llevan a cabo con una interfaz DDE (Intercambio dinámico de datos, del inglés *Dynamic Data Exchange*) que conecta el microcontrolador con la plataforma IoT online EasySensing, una plataforma creada previamente por los autores.

Así mismo, se han llevado a cabo estudios para edificios en concreto, como en (Lestari, Wahyono, & Fadlika, 2018) o en (Choi et al., 2017) o para una infraestructura para tratar edificios en general, combinando IoT y BIM (Modelado de información de construcción, del inglés *Building information modeling*) (Bottaccioli et al., 2017).

En el primer estudio, se desarrolla un prototipo de monitorización en tiempo real del consumo de energía en un sistema HVAC de un edificio. El prototipo está

basado en una Raspberry Pi 2, un ordenador de placa reducida. El esquema de comunicación está basado en una arquitectura de publicación-servidor-subscriptor, donde la Raspberry Pi 2 obtiene los datos de los sensores y tras su procesado los publica mediante HTTP en un servidor web en forma de página web accesible desde otros dispositivos.

En (Choi et al., 2017), se implementa un sistema inteligente de control del sistema HVAC para disminuir el consumo necesario en pos de acercarse más a edificios con consumo energético neto nulo en un futuro. Esto se consigue mediante dispositivos IoT dispuestos en cada zona, que monitorizan independientemente y controlan el sistema HVAC de cada una, de manera que se cumplan parámetros de confort, manteniendo al mínimo el consumo eléctrico. El sistema está diseñado *ad hoc* para un centro de personas mayores, usando una pasarela IoT física que recoge los datos de los sensores y actúa como servidor.

En (Bottaccioli et al., 2017) se presenta una arquitectura de software para la administración y simulación de edificios, combinando BIM con el internet de las cosas y modelos geográficos.

El sistema permite combinar los datos del edificio con datos geográficos y medidas de sensores en tiempo real dentro del edificio. A partir de toda esta información, el software permite hacer simulaciones basadas en datos reales actualizados, monitorizar y evaluar el comportamiento energético del edificio. Esto es muy interesante, ya que este tipo de simulaciones, que se suelen realizar con un modelo hipotético de la media anual meteorológica, se puede realizar con datos reales, incrementando la precisión de los resultados.

En cuanto a estos trabajos, en (Ruano et al., 2018) se centran específicamente en el modelo predictivo, y en (Lestari et al., 2018) los resultados de la solución propuesta son difíciles de monitorizar, ya que solo son accesibles desde la red local.

Las soluciones anteriores usan un sistema más bien caro en comparación, por ejemplo, con (Serra, Pubill, Antonopoulos, & Verikoukis, 2014). En este último artículo se propone un dispositivo IoT de bajo coste que controla, mediante un algoritmo predictivo, ciertos actuadores de un sistema HVAC doméstico. La monitorización del sistema permite controlarlo remotamente, disminuyendo el consumo eléctrico de una habitación. Utiliza una WSN (red de sensores

inalámbricos, del inglés *Wireless Sensor Network*) compuesta por varios Z1 de Zolertia, un microcontrolador con varias entradas y capacidades WiFi. Unos actúan como sensor inalámbrico mientras que otro recoge las señales y las redirige a un ordenador con Matlab, donde se lleva a cabo el procesado y se publica en un servidor web.

Por último, también hay trabajos centrados en monitorizar la calidad del aire, como en (Rinaldi, Flammini, Tagliabue, & Ciribini, 2019).

En cuanto al microcontrolador ESP8266, usado en el presente proyecto, ha sido utilizado con éxito en (Kodali & Mahesh, 2016; Saha & Majumdar, 2017; Wan, Song, & Cao, 2019) como sistema de bajo consumo, con la gran ventaja de poder mostrar la información remotamente, ya que dispone de capacidades WiFi nativas.

En (Wan et al., 2019) se desarrolla un sistema de bajo coste de administración de un invernadero controlado remotamente. El sistema está basado en un microcontrolador ESP8266 y recoge valores en tiempo real de CO_2, temperatura y humedad del aire y de la tierra, en las cuales se basa para luego corregir, si es necesario, tales parámetros mediante varios actuadores conectados al microcontrolador. Los datos se envían mediante WiFi a una plataforma IoT.

En (Saha & Majumdar, 2017) se presenta un sistema de monitorización de temperatura en un datacenter. El ESP8266 envía por WiFi los datos recogidos por sus sensores a una plataforma IoT, donde se presentan gráficamente a través de internet y son accesibles desde cualquier lugar.

Pero en ambas soluciones la comunicación final con el usuario es mediante plataformas de pago como Ubidots (Saha & Majumdar, 2017) o servicios cerrados como Thinkspeak o Bylnk (Wan et al., 2019), los cuales ofrecen versiones gratuitas solo para pequeños proyectos.

En definitiva, la mayoría de trabajos realizados anteriormente son soluciones para problemas muy específicos, una buena parte de estos están centrados en estudiar el consumo de sistemas HVAC de edificios concretos. En otros, simplemente se usa el sistema IoT para validar modelos predictivos de control para mantener el confort en un edificio.

Todas estas soluciones tienen un precio más bien elevado, dado que se utilizan microcontroladores que además requieren de módulos adicionales para dotarles de conectividad inalámbrica. El coste total de estos sistemas se incrementa sustancialmente respecto a usar, por ejemplo, un microcontrolador de bajo coste con conectividad inalámbrica integrada. Esto ha llevado a elegir el microcontrolador ESP8266 para la realización del presente trabajo, ya que dispone de soporte WiFi integrado.

Finalmente, también se ha contrastado que la mayoría de trabajos y en especial los realizados con el ESP8266 utilizan plataformas IoT ya desarrolladas, y que estas son de pago y/o muy cerradas, en el sentido que no permiten extraer los datos recogidos para un procesamiento más a fondo de estos, por lo tanto, sería interesante diseñar una plataforma más abierta.

En los próximos capítulos se dará un marco teórico para explicar mejor el sistema de comunicaciones y la arquitectura del sistema: de dónde viene la fiabilidad de los protocolos escogidos y cómo se conectará el microcontrolador a la base de datos.

Más adelante se hablará de la elección del microcontrolador, de su programación y de la elección de la base de datos y el software de visualización.

4 Comunicaciones y protocolos

Para reducir su complejidad, las redes informáticas se suelen organizar en capas. Cada capa es un nivel de abstracción que oculta sus detalles, ofreciendo a las capas superiores ciertos servicios a través de interfaces (Figura 3). Dentro de una red, la comunicación entre capas se lleva a cabo a través de los protocolos de capa, los cuales son estándares que definen unas reglas unívocas que permiten la comunicación correcta entre los sistemas informáticos de la red.

La información viaja de las capas superiores (de aplicación), codificada según los protocolos de cada capa, hacia las capas inferiores (de comunicación física) a través de interfaces que le añaden metadatos, y llega al destinatario, donde vuelve a subir a las capas superiores usando otra vez las interfaces. Este proceso se llama encapsulamiento de la información, ilustrado en la Figura 2.

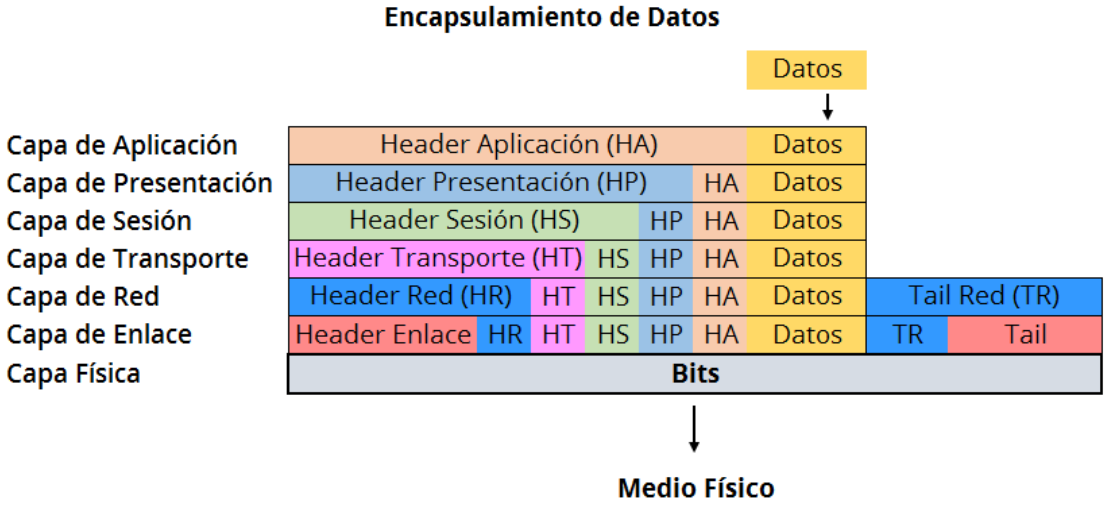

Figura 2: Ejemplo de encapsulamiento de datos, usando el modelo OSI

Para la capa de aplicación la comunicación es aparentemente directa, resultado de la abstracción de las capas inferiores. La abstracción por capas es vital para reducir el desarrollo necesario en aplicaciones. Solo es necesario abordar el diseño de la comunicación con la capas inmediatamente inferior y superior para obtener un diseño final funcional.

El conjunto de capas usadas y protocolos de estas definen la arquitectura de una red. En el próximo capítulo se definirá la arquitectura del proyecto en base a lo expuesto en el presente capítulo.

Hay varios modelos que describen las capas necesarias en una red, el más estandarizado es el modelo OSI (modelo de interconexión de sistemas abiertos del inglés, *Open System Interconnection*) que se explicará a continuación.

4.1 Modelo OSI

El modelo OSI es un modelo conceptual de red muy general, no ha sido muy usado debido a que describe siete capas que son difíciles de asignar en ciertas arquitecturas actualmente en uso, pero aún así es muy explicativo. Fue publicado por la ISO (Organización Internacional de Normalización, del inglés International *Organization for Standardization*) en 1984 como estándar ISO 7498.

La intención de este modelo es crear una capa para cada nivel de abstracción necesario, con la función de cada capa bien definida, permitiendo la creación de protocolos estandarizados y minimizando el flujo de información entre capas.

No se considera una arquitectura de red ya que no define los protocolos ni los servicios de cada capa, solo explica su función. (Tanenbaum & Wetherall, 2011)

En la Figura 3 se muestra el esquema del modelo, para continuar con una breve definición de sus funciones empezando por las capas más bajas.

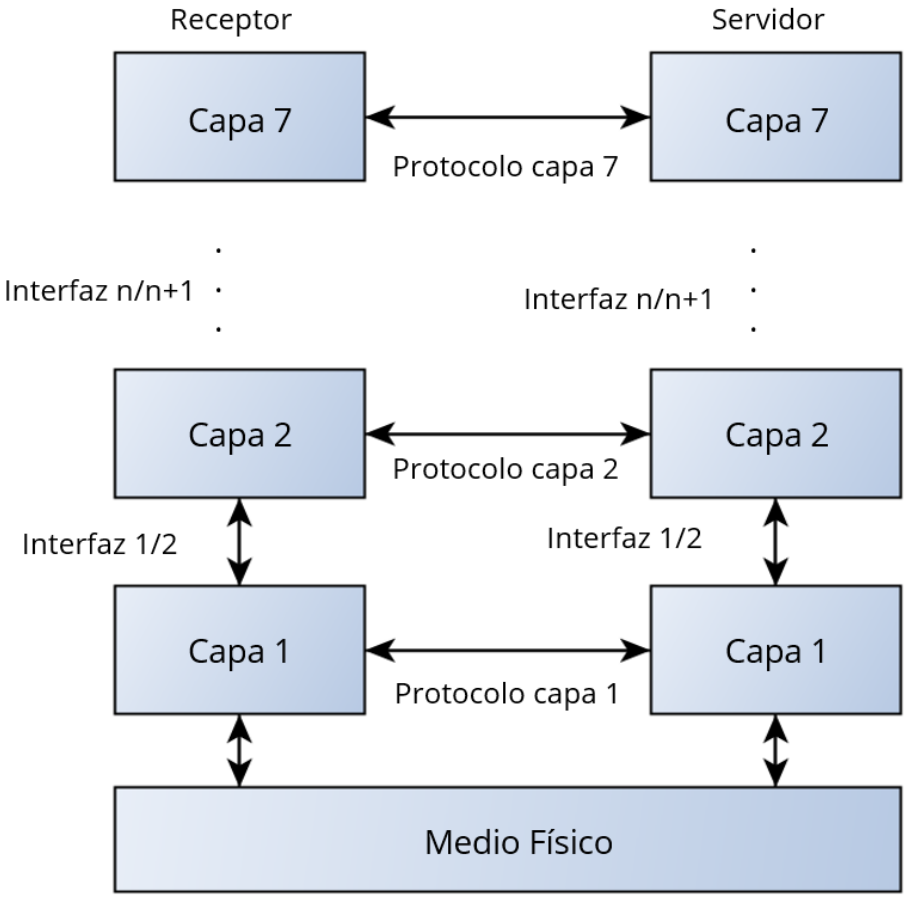

Figura 3: Capas, interfaces y protocolos del modelo OSI

4.1.1 Capa física

La capa física proporciona los procedimientos mecánicos, eléctricos y funcionales para activar, desactivar y mantener las comunicaciones. Tiene que definir características tales como el medio físico de transmisión, el tipo de señal eléctrica, la cadencia de transmisión de los datos, o el tipo de comunicación (en un solo sentido, *simplex* o en ambos sentidos, *dúplex*).

4.1.2 Capa de enlace de datos

Proporciona los procedimientos para transformar la transmisión recibida de la capa física en una transmisión sin errores, dividiendo en tramas ordenadas los datos recibidos, transmitiendo estas tramas secuencialmente. Además, establece un control de flujo para evitar que el destinatario se desborde con los datos recibidos (reciba más de los que puede procesar).

4.1.3 Capa de red

Se ocupa del enrutamiento de los datos, es decir, escoger el mejor camino entre un remitente y un destinatario de una red para enviar los paquetes que componen los datos. También debe encargarse de como redirigir paquetes provenientes de otras redes.

La calidad del servicio también depende de esta capa, la cual ha de intentar reducir los retrasos y la latencia. Finalmente, su protocolo puede o no integrar transporte fiable, en el sentido de que si es fiable se espera una respuesta del destinatario o *acknowledgement* una vez este recibe los datos y si no lo es, no se espera tal respuesta.

4.1.4 Capa de transporte

Esta es la capa más baja sobre la cual la comunicación es directamente entre el destinatario y el remitente, contrariamente de las anteriores, donde la comunicación era entre la red y uno de ellos. Es por este motivo que se puede interpretar como independiente de las inter-redes (red de redes heterogéneas o no) que utiliza. Su función es pasar los datos a la capa de red, por orden o no, y fragmentándolos si es necesario.

Los protocolos TCP (protocolo de control de transmisión, del inglés *Transmission Control Protocol*) y UDP (protocolo de datagramas de usuario, del inglés *User Datagram Protocol*) pertenecerían a esta capa, aunque no son propios del modelo OSI.

4.1.5 Capa de sesión

La capa de sesión, entre otras funciones, permite establecer una sincronización para lograr reanudar una conversación fallida entre remitente y destinatario en el mismo punto dónde se dejó. También regula el diálogo para saber quién recibe y quién transmite en todo momento y habilita una operación *dúplex*, *semi-duplex* o *simplex*.

Estas funciones pueden ser reemplazadas por otras capas y en otros modelos se omite debido a que puede verse como superflua.

4.1.6 Capa de presentación

La capa de presentación se encarga de traducir, según ciertos estándares, la información recibida de las capas de aplicación a un formato transferible por la capa de red, y viceversa: pasar de este formato transferible a uno comprensible por la aplicación. En definitiva, su objetivo es obtener una estructura comprensible para el destinatario, en el ámbito de la sintaxis y semántica de los datos.

Al igual que la capa de sesión, otros modelos la omiten, ya que se puede considerar que la aplicación/programa se encarga de estas funciones.

4.1.7 Capa de aplicación

La capa de aplicación permite acceder a los servicios de las capas inferiores. Los usuarios no suelen interactuar directamente con la capa de aplicación, sino accediendo a programas/aplicaciones que la usan.

El protocolo HTTP, que es la base de la WWW (red informática mundial) del inglés *World Wide Web*) se clasificaría como protocolo de esta capa.

4.2 Modelo TCP/IP

También llamado suite de protocolos de internet, es un modelo conceptual de red que, al contrario del modelo OSI, ha sido muy usado ya que los protocolos que emplea fueron aplicados antes de la definición formal del modelo.

Se pueden establecer equivalencias no estrictas con el modelo OSI, tal y como se muestra en la Tabla 1. Es visible que el modelo TCP/IP no define tantas capas, omitiendo las de sesión y presentación, como muchos otros modelos, y la equivalente a la capa física.

En el presente trabajo se emplearán ciertos protocolos pertenecientes a este modelo, por lo tanto, se considera necesario hacer mención previamente a las capas del modelo y sus protocolos correspondientes.

TCP/IP	OSI
Capa de acceso al medio	1. Capa física
	2. Capa de enlace de datos
Capa de internet	3. Capa de red
Capa de transporte	4. Capa de transporte
Aplicación	5. Capa de sesión
	6. Capa de presentación
	7. Capa de aplicación

Tabla 1: Comparación de los modelos de referencia

4.2.1 Capa de acceso al medio

Técnicamente no es una capa ya que más bien describe la interfaz necesaria para obtener una comunicación no basada en la conexión (no espera respuesta ni pide empezar la transmisión de datos).

4.2.2 Capa de internet

Se corresponde con la capa de red del modelo OSI. En ella se define el protocolo de internet o IP (del inglés *Internet Protocol*). El protocolo de internet, junto con el ICMP (Protocolo de control de mensajes de internet, del inglés *Internet Control Message Protocol)* se encargan del enrutamiento de los paquetes de datos. El ICMP se encarga de transmitir mensajes de error y de información operacional sobre la comunicación, pero no de transmitir datos.

4.2.3 Capa de transporte

Tal y como su homónimo en el modelo OSI, se encarga de permitir la comunicación entre el remitente y el destinatario, así como la comunicación con la capa de internet. También es independiente de la red. Cabe destacar dos protocolos fundamentales del modelo: UDP y TCP.

El primero es usado en aplicaciones donde prima el rendimiento, ya que UDP no incluye técnicas de gestión de tráfico, y es un protocolo no fiable y no basado en la comunicación (no garantiza la llegada de los datos, y no pide formar una conexión al destinatario previamente).

El segundo es fiable y basado en la comunicación, y permite a los datos llegar al destinatario secuencialmente. También incluye técnicas de gestión de tráfico, procurando de no desbordar al destinatario si el remitente es más rápido.

4.2.4 Aplicación

Como hemos visto anteriormente, se prescinde de las capas de sesión y presentación, las cuales son de poco uso y son directamente aplicadas en los programas/aplicaciones. Así, solo queda la capa de aplicación, la cual contiene varios protocolos muy utilizados, en especial HTTP (Protocolo de transferencia de hipertexto, del inglés *Hypertext Transfer Protocol*), el cual veremos más a fondo en las próximas secciones ya que es de especial interés para el proyecto.

4.3 Protocolos utilizados

Como se verá en el próximo capítulo, la comunicación entre el microcontrolador y la base de datos del proyecto será realizada mediante protocolo HTTP.

En la Figura 4, se muestran los protocolos utilizados en cada capa según el modelo TCP/IP, posteriormente se explican brevemente.

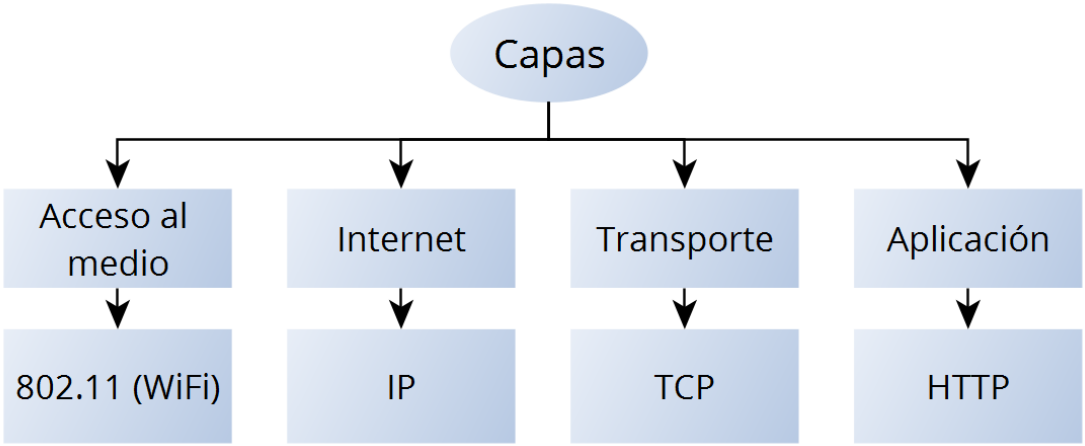

Figura 4: Protocolos del modelo TCP/IP usados en cada capa en el diseño del proyecto

4.3.1 Protocolos IEEE 802.11

En este conjunto de protocolos se describe la implementación de una red local inalámbrica, son el estándar usado en la mayoría de dispositivos inalámbricos hoy en día. Es un subconjunto de los protocolos LAN (red de área local, del inglés *Local Area Network)* IEEE 802. Por su función, en el modelo OSI pertenece a la capa física y a la de enlace de datos.

En cuanto a la capa física, el principio físico se basa en la modulación y demodulación de ondas portadoras (normalmente de 2.4 GHz o 5 GHz), que transmiten los datos mediante diferentes técnicas según el protocolo específico. Actualmente, los más usados son 802.11n/g/b y 802.11ac, siendo este último el más moderno que busca incrementar la velocidad de transmisión, así como reducir las pérdidas de datos debidas a las interferencias entre equipos (colisiones).

La función de enlace de datos se lleva a cabo encapsulando los datos: añadiendo una cabecera formando una trama, la cual contiene una cabecera con datos relevantes a la comunicación, como la dirección MAC (control de acceso al medio, del inglés *Media Access Control*) del destinatario y el remitente, y otros campos de control. A diferencia de los demás encapsulamientos que veremos en las demás capas, también cierra el encapsulamiento con un fin de trama.

4.3.2 Protocolo de internet (IP)

El protocolo de internet o IP fue diseñado específicamente para interconectar redes heterogéneas, permitiendo la comunicación a través de esta red de redes, sin importar sus diferencias en cuanto tecnologías usadas, por ejemplo, conectar redes inalámbricas con alámbricas. Su principal función es el enrutamiento: escoger el camino que los paquetes de datos deben tomar para llegar a su destinatario, y lo logra mediante un sistema de direcciones IP.

La especificación técnica original esta publicada en RFC 791 (petición de comentarios, del inglés *Request For Comments*). Pertenece a la capa de red en el modelo OSI.

Para lograr su cometido, este protocolo encapsula una vez más los datos añadiendo una cabecera (Figura 5).

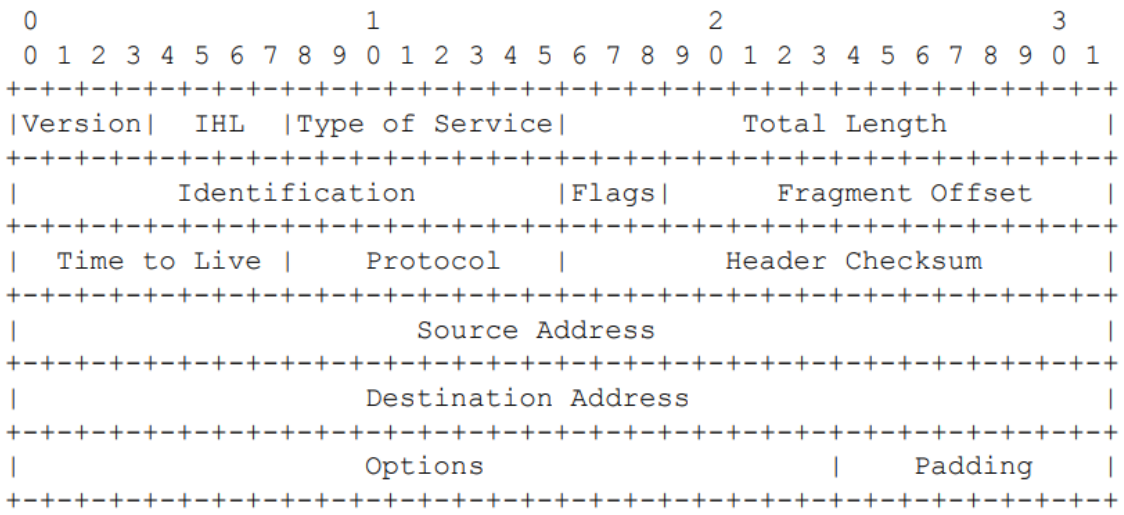

Figura 5: Cabecera IP original, 32 bits de ancho por cada línea (Postel & others, 1981a)

La cabecera añade varios campos, entre ellos podemos destacar:

Versión del protocolo (*Version*): Dado al incremento de usuarios de internet el protocolo usado en la actualidad (IPv4) se esta quedando sin direcciones para asignar, ya que solo puede asignar 2^{32} direcciones en bloques, y por lo tanto hay una nueva versión (IPv6) que se esta implementando paulatinamente, que entre otras mejoras permite asignar 2^{128} direcciones.

El campo de protocolo permite dar a conocer, mediante a una numeración estandarizada, el protocolo que esta encapsulado dentro de la IP, necesario para saber que protocolo ha de encargarse de los datos p.ej. TCP o UDP.

De vital importancia son los campos de direcciones (*Source/Destination adress*) que contienen la dirección IP del destinatario y el remitente. Estos campos son traducidos a términos de dirección MAC gracias al ARP (protocolo de resolución de direcciones, del inglés *Address Resolution Protocol*), necesario para el funcionamiento de la capa inferior.

Finalmente, incluye también un campo de verificación de datos (*checksum*) y otro de opciones que han caído en desuso dado que se sus funciones se implementan en otras capas.

4.3.3 Protocolo de control de transmisión (TCP)

El protocolo de control de transmisión (TCP del inglés *Transmission Control Protocol*) se definió formalmente en 1981 en el RFC 793. Pertenece a la capa de transporte en el modelo OSI y en el TCP/IP, al cual le da parte del nombre debido a su importancia.

Es uno de los protocolos más importantes de internet, fue diseñado como un protocolo fiable y robusto para comunicar información de punto a punto sobre una red de redes no fiable. Para conseguirlo encapsula los paquetes de la capa de red añadiendo las cabeceras TCP la cual permite añadir varias funciones mediante sus campos. A continuación, se presentan algunos de los campos. La estructura de la cabecera se puede ver en la Figura 6.

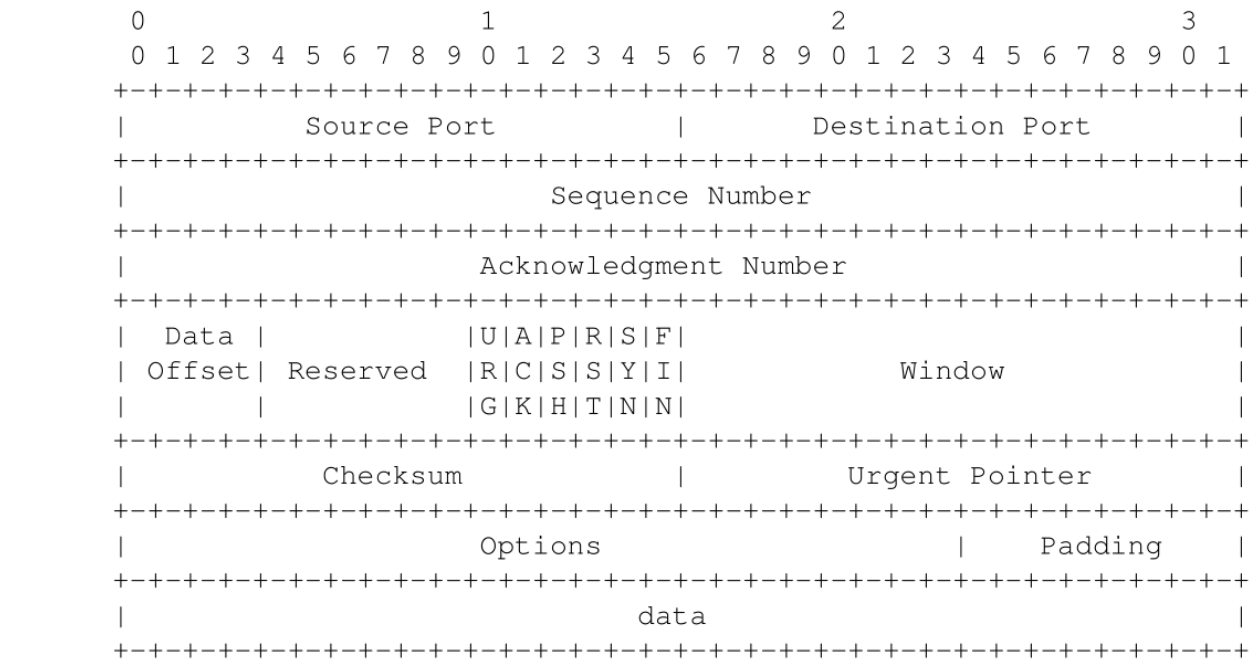

Figura 6: Cabecera TCP original, 32 bits de ancho por cada línea (Postel & others, 1981b).

Puertos de origen y destinación (*Ports*), para implementar comunicaciones para aplicaciones identificadas con estos puertos.

Números de secuencia para mantener el orden de los paquetes al llegar al receptor, ya que durante la comunicación se pueden perder paquetes o retrasar, perdiendo estos el orden de llegada correcto.

Establecimiento de la conexión mediante negociación en tres pasos, durante la cual el cliente primero manda una señal al servidor, que responderá con una señal de asentimiento (o negación si la rechaza), el cliente enviará una señal de asentimiento entonces y otro envío con los datos (*Data*).

Los asentimientos (*Acknowledgment*) y señales (*Sequence number*) son contabilizados según el número de bits enviados, lo que permite asegurarse de que la comunicación se realiza correctamente.

El protocolo también cuenta con otras opciones más avanzadas relacionadas con el control de flujo, sumas de verificación (*Checksum*) para verificar la integridad de los datos etc. Además, cuenta con otras mejoras en seguridad y rendimiento que se han hecho con el paso de los años.

4.3.4 Protocolo de transferencia de hipertexto

El protocolo de transferencia de hipertexto (HTTP del inglés *Hypertext Transfer Protocol*) está definido en RFC 2616. Pertenece a la capa de aplicación en el modelo OSI.

Es un protocolo de petición-respuesta que suele correr sobre TCP, aunque próximamente, para su versión 3 (HTTP/3) se hará correr sobre UDP debido al aumento de rendimiento posible. Por convenio se usa el puerto 80 para establecer esta comunicación, o el 443 en caso de HTTPS (HTTP Seguro, del inglés *Secure*).

Al contrario de los protocolos anteriores, la línea de petición y las cabeceras se transmiten directamente en ASCII (Código Estándar Estadounidense para el Intercambio de Información, del inglés *American Standard Code for Information Interchange*), por lo cual es de fácil lectura para personas.

En la Figura 7 se desglosa un ejemplo de comunicación mediante HTTP a modo ilustrativo. En la petición se transmite cierta información, como a quién esta dirigida, el método (que define la operación a realizar) o la versión del protocolo a usar. Después de esta primera línea, se envían las cabeceras: metadatos como por ejemplo el tipo de datos que se aceptan o quién hace la petición. Después de una línea en blanco se envían datos en el cuerpo del mensaje si el método los requiere. La respuesta contiene campos similares.
En la Tabla 2 se muestran los métodos más comunes, y su descripción.

El protocolo HTTP es famoso por ser el protocolo base de la WWW, pero no es exclusivo de esta, de hecho, se usa en multitud de aplicaciones y será utilizado en este proyecto para enviar los datos de los sensores a la base de datos, ya que la API de la base de datos usa HTTP.

Método	Descripción
GET	Solicita el recurso especificado
HEAD	Solicita las cabeceras del recurso, sin el cuerpo/contenido
POST	Pide almacenar el contenido de la petición en la URI especificada
PUT	Pide almacenar el contenido de la petición en la URI especificada, si el recurso ya existe, lo sobrescribe
DELETE	Elimina el recurso especificado

Tabla 2: Métodos más comunes de HTTP.

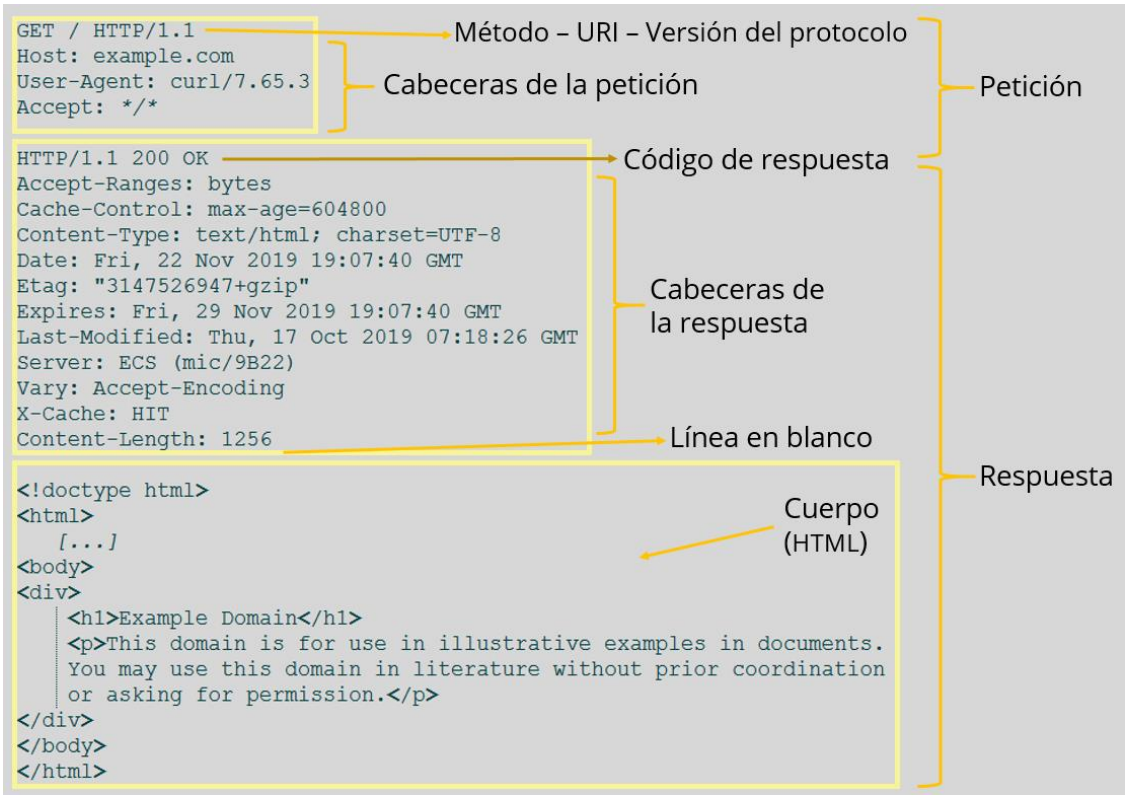

Figura 7: Ejemplo de petición y respuesta del protocolo HTTP

5 Diseño del sistema

A continuación, se desarrollará en detalle el diseño del sistema, empezando por la justificación de InfluxDB como base de datos y Grafana como software de visualización a partir de las necesidades del sistema. Se continuará con el diseño de la arquitectura del sistema enumerando y describiendo sus elementos y sus comunicaciones.

Finalmente se justificará el hardware utilizado y se comentará la programación del microcontrolador e integración con las plataformas.

5.1 Plataformas utilizadas

Dependiendo de la aplicación, una base de datos puede no ser necesaria, ya que el mismo microcontrolador puede suplir parcialmente su función actuando como un servidor web y directamente mostrar la información en gráficos dinámicos. Aunque el microcontrolador escogido en el presente trabajo tiene una capacidad limitada como servidor web, se podría haber escogido otras alternativas más potentes como veremos en los próximos capítulos.

No obstante, la razón por la que se ha escogido utilizar un diseño que cuenta con una base de datos y una herramienta de visualización para la base de datos es la ventaja de que estas herramientas ofrecen funcionalidades muy potentes con un coste de implementación muy bajo.

La base de datos, por sí sola, puede servir para más tarde hacer un post-proceso de la información recogida. Además, permite manejar fácilmente la información: mediante peticiones avanzadas con varias condiciones y mediante las funcionalidades que ofrece, como políticas de retención de datos, las cuales permiten borrar los datos a partir de cierto tiempo o comprimirlos.

Por otro lado, una herramienta de visualización permite producir *dashboard*, tableros de control con gráficos interactivos, de una manera sencilla, ya que estas plataformas están altamente integradas con multitud de bases de datos para facilitar el desarrollo. También suelen contar con funcionalidades a parte de crear los *Dashboards*, tal como sistemas de alarmas o post-procesamiento de los datos.

Adicionalmente, con este diseño se obtiene una estructura fácilmente escalable, debido a que una vez hecha la estructura para un microcontrolador es sencillo añadir otro a la base de datos, y la alta integración de las plataformas de visualización permite generar nuevos *dashboard* de manera sencilla.

A continuación, se justifica la elección de las plataformas finalmente escogidas: InfluxDB como base de datos y Grafana como herramienta de visualización.

5.1.1 InfluxDB

InfluxDB es una base de datos de código abierto NoSQL (no SQL, del inglés *non SQL*) que pertenece a las TSDB (Base de datos temporal, del inglés *Time Series Database*). Trabaja con un sistema HTTP *push* en el que los datos son subidos a la base de datos en vez de esta solicitarlos, con lo cual el microcontrolador ha de ser programado para enviar los datos cada cierto tiempo mediante HTTP.

Las TSDB son bases de datos optimizadas para trabajar con valores asociados a un registro de tiempo, en comparación a bases de datos que emplean un modelo más clásico del tipo SQL (lenguaje de consulta estructurada, del inglés *Structured Query Language*).

Esta optimización permite una escritura y lectura de los datos muy rápida, además de contar con funcionalidades avanzadas para tratar el ciclo de vida la gran cantidad de datos que puede recoger y mantener un uso del almacenamiento racional. Por estos motivos, debido a que se van a recoger este tipo de valores, se ha decidido trabajar con una TSDB.

Hay varias TSDB disponibles, tanto de código abierto como comerciales. Para facilitar el desarrollo del trabajo solo se han considerado TSDB de código abierto, por ejemplo, buscando entre las más utilizadas, InfluxDB, Kdb+, Prometheus, Graphite u OpenTSDB.

Estas se comparan a continuación en la Tabla 3 según varias características que se han considerado fundamentales para la realización del proyecto: la API, la posibilidad de usar *tags*, la capacidad de *Downsampling,* el uso de técnicas para almacenamiento duradero, y la granularidad del almacenamiento. Adicionalmente se ha tenido en cuenta el soporte comercial.

TSDB	API	*Tags*	*Downsampling*	Almacenamiento duradero	Granularidad de almacenamiento	Soporte comercial
Graphite	HTTP GUI, HTTP, JSON	Sí	No	Sí	1 s	No
InfluxDB	Collectd, CLI, Graphite, HTTP, OpenTSDB, UDP	Sí	Sí	Sí	1 ns	Sí
KDB+	CLI	No	No	Sí	1 ns	Sí
OpenTSDB	CLI, Azure, HTTP GUI, HTTP, REST, JSON, Kafka, RabbitMQ, S3, Spritzer	Sí	Sí	No	>1 ms	No
Prometheus	CLI, HTTP GUI, HTTP	Sí	Sí	No	1 ms	No

Tabla 3: Tabla comparativa de las TSDB de código abierto más utilizadas, información extraída parcialmente de (Bader, Kopp, & Falkenthal, 2017)

Se consideran las APIs incluidas sin necesidad de usar software de terceros, lo cual resulta en una implementación más limpia. Es importante que la base de datos a utilizar tenga una interfaz HTTP ya que es el modo de comunicación más sencillo para el microcontrolador, por lo que directamente se puede descartar *KDB+*.

En cuanto a los *tags,* se ha tenido en cuenta que la base de datos pueda asignar etiquetas a los datos recogidos, lo cual es de vital importancia para escalar el sistema ya que permite identificar la fuente de los datos, por ejemplo, datos de distintos microcontroladores o sensores.

El *Downsampling* se define como la capacidad de una base de datos para ajustar los datos cuando se requiere un resultado en un periodo de tiempo más largo. Por ejemplo, si se realiza una petición de los datos a lo largo de un mes, la base de datos puede dar las medias de cada día, lo cual es más eficiente que dar los datos de cada segundo. Ni Graphite ni KDB+ disponen de *Downsampling* nativamente, por lo que son descartadas por este motivo.

En la columna de almacenamiento duradero se ha incluido: el uso de técnicas de compresión, la retención de datos solo durante un periodo establecido, y/o la disminución de la resolución de los datos para periodos anteriores a cierto punto. Esto es conveniente a fin de reducir el volumen de los datos para el sistema a largo plazo. Sin ninguna de estas técnicas, se podrían generar grandes cantidades de datos que pueden acaparar muchos recursos. Tanto Prometheus como openTSDB no disponen de estas técnicas, por lo cual han sido descartadas.

La granularidad del almacenamiento hace referencia a la mínima diferencia de tiempo que la base de datos acepta entre medidas. Por ejemplo, en el caso de KDB+ e InfluxDB, la base de datos acepta medidas cada nanosegundo.

Aunque no es requerido para el presente trabajo, el hecho de tener soporte comercial en forma de soporte técnico es importante en el momento de abordar proyectos más grandes. Este principalmente se ofrece en las versiones comerciales de pago de KDB+ e InfluxDB, que, por ejemplo, en el caso de InfluxDB, ofrecen características interesantes como *clustering*.

Finalmente se ha elegido InfluxDB como base de datos. Junto al hecho de que ofrece una API HTTP, *tags*, *downsampling,* técnicas de almacenamiento duradero y una buena granularidad, características que las demás TSDB no ofrecen juntas, también se ha escogido por su contrastada eficiencia. Esta eficiencia se debe a que es una TSDB construida desde cero, no a partir de una base de datos normal, y por lo tanto la optimización como TSDB es su objetivo principal.

InfluxDB también dispone de una gran base de usuarios, y está clasificada como la TSDB más popular según (DB-Engines, n.d.). Finalmente, cuenta con una documentación extensa, lo cual es una gran ventaja ya que facilita el aprendizaje.

5.1.2 Grafana

Grafana es una herramienta de visualización de TSDB de código abierto. Está preparada para trabajar directamente con multitud de bases de datos, tanto SQL como NoSQL. Funciona haciendo peticiones HTTP a dichas bases de datos.

Las herramientas de visualización permiten monitorizar y observar la información almacenada en una base de datos. Algunas disponen de funcionalidades como alertas según condiciones, gráficos interactivos, exploración de datos o cierto post-proceso de los datos, como es el caso de Grafana.

Existen varias herramientas de visualización como Kibana, Metabase, Graphite-web, NetData, Chronograf o Grafana. Las últimas dos opciones están altamente integradas con InfluxDB, siendo Chronograf parte de InfluxData, la empresa que gestiona InfluxDB, y Grafana un *partner* de InfluxData desde 2014. Por su parte, la primera versión de Chronograf se publicó en 2015.

Kibana no es compatible con InfluxDB, ya que está concebida para trabajar solo con Elasticsearch. Metabase no soporta InfluxDB y Graphite-web está diseñada en torno a su propia base de datos, Whisper. NetData se ha descartado al no ser compatible con Windows, y no ofrecer ningún beneficio adicional respecto a Chronograf o Grafana.

Entre las dos herramientas de visualización restantes, Chronograf está más enfocada a la exploración de datos presentes en InfluxDB mediante el lenguaje de procesamiento *FLUX,* y Grafana más orientada a la monitorización de los datos una vez se encuentran relaciones interesantes entre ellos. Ambas herramientas son de

código abierto y comparten la visualización de los datos en forma de paneles organizados en *dashboards*.

La principal razón por la que se ha escogido Grafana en detrimento de Chronograf es por su gran base de usuarios. Debido a esto, junto con que lleva más tiempo en el mercado, y su uso esta más extendido que el de Chronograf.

Grafana además posee un sistema de alertas más completo y sencillo de utilizar, y permite utilizar varias bases de datos, no solo InfluxDB, esto resulta en una solución más escalable y versátil en futuras implementaciones.

5.2 Arquitectura del sistema

Teniendo en cuenta los requerimientos expuestos en la introducción, así como las tendencias y carencias vistas en el estado del arte, se plantea la arquitectura de la Figura 8.

La arquitectura incluye el ESP8266, que como se ha comentado anteriormente es un microcontrolador económico que cumple los requerimientos específicos del proyecto. Los sensores, como se verá más adelante, se conectarán a este a través de un ADC (conversor analógico digital, del inglés *Analog to Digital Converter*), cuya función principal es ampliar las entradas analógicas disponibles al microcontrolador. El ADC se conectará mediante SPI (Interfaz periférica serial, del inglés *Serial Peripheral Interface*), las entradas adicionales permitirán utilizar el sistema para un amplio rango de aplicaciones en un futuro.

El microcontrolador será el encargado de formatear los datos obtenidos por los sensores, así como de hacer un pre-procesado de estos, si fuese necesario. Así mismo se conectará con la base de datos (InfluxDB) mediante una API (interfaz de programación de la aplicación, del inglés *Application Programming Interface*) HTTP a través de WiFi.

La base de datos, que se aloja en un ordenador, también mediante una API HHTP, se comunica con la herramienta de visualización (Grafana), que mediante distintos *dashboard* muestra la información recogida y puede añadir cierto procesado de datos, así como alarmas. El servicio web de la herramienta de visualización también se ejecuta en el mismo ordenador, aunque esto no es necesario.

La herramienta de visualización es accesible a través de internet y permite acceder a ella desde cualquier navegador, actuando como un servicio web y ofreciendo, por lo tanto, acceso a multitud de dispositivos.

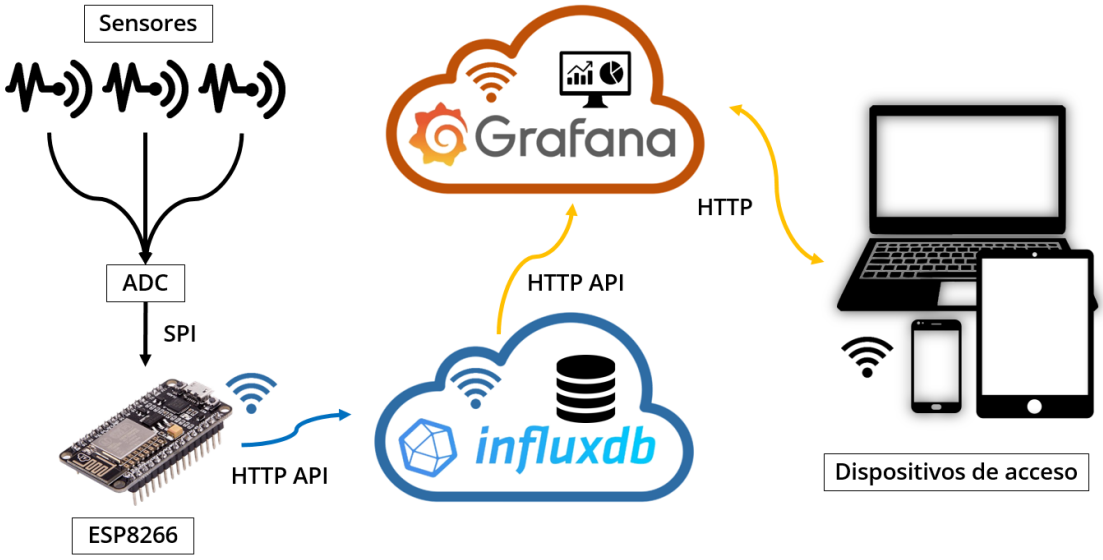

Figura 8: Diagrama de la arquitectura propuesta para el sistema

Esta estructura es escalable ya que la base de datos puede recoger información de varios microcontroladores a la vez, fiable gracias a los protocolos escogidos, y robusta debido a la programación del microcontrolador, que ante situaciones adversas debe reaccionar de una manera adecuada; por ejemplo, debe intentar reconectarse a una red en caso de pérdida de conexión.

5.3 Red y comunicaciones

En este apartado se hablará de como se lleva a cabo la comunicación entre los elementos del sistema vistos previamente en la Figura 8.

5.3.1 SPI

SPI es una interfaz de comunicación serial síncrona, usada para transmisión de datos en distancias cortas, normalmente en circuitos integrados. Implementa una comunicación maestro-esclavo dúplex total (se puede transmitir y recibir a la vez y la comunicación es independiente), permitiendo altas velocidades de transmisión.

Para esta interfaz se requieren de cuatro buses, lo cual es su principal desventaja:

- SCLK: La señal de reloj generada por el maestro para habilitar la comunicación síncrona.
- MOSI: Salida de datos del maestro, entrada de datos del esclavo (*Master Output Slave Input*).
- MISO: Entrada de datos del maestro, salida de datos del esclavo (*Master Input Slave Output*).
- SS: Selección del dispositivo esclavo (*Slave Select*). Permite poder usar varios dispositivos esclavos con un solo maestro sin añadir más buses, ya que se pueden encadenar los esclavos entre ellos.

En el protocolo SPI se conectan los registros de desplazamiento del maestro y el esclavo circularmente utilizando las líneas MISO y MOSI. Los datos se transfieren entonces síncronamente según la señal de reloj hasta que los registros de desplazamiento se han intercambiado completamente, momento en que se lee y se pasa al siguiente ciclo (Figura 9). Se intercambia por lo tanto un bit por cada flanco de reloj hasta intercambiar los registros que tienen una longitud determinada (normalmente de uno o dos bytes); esto posibilita una transmisión muy rápida y lo hace más atractivo que otras interfaces.

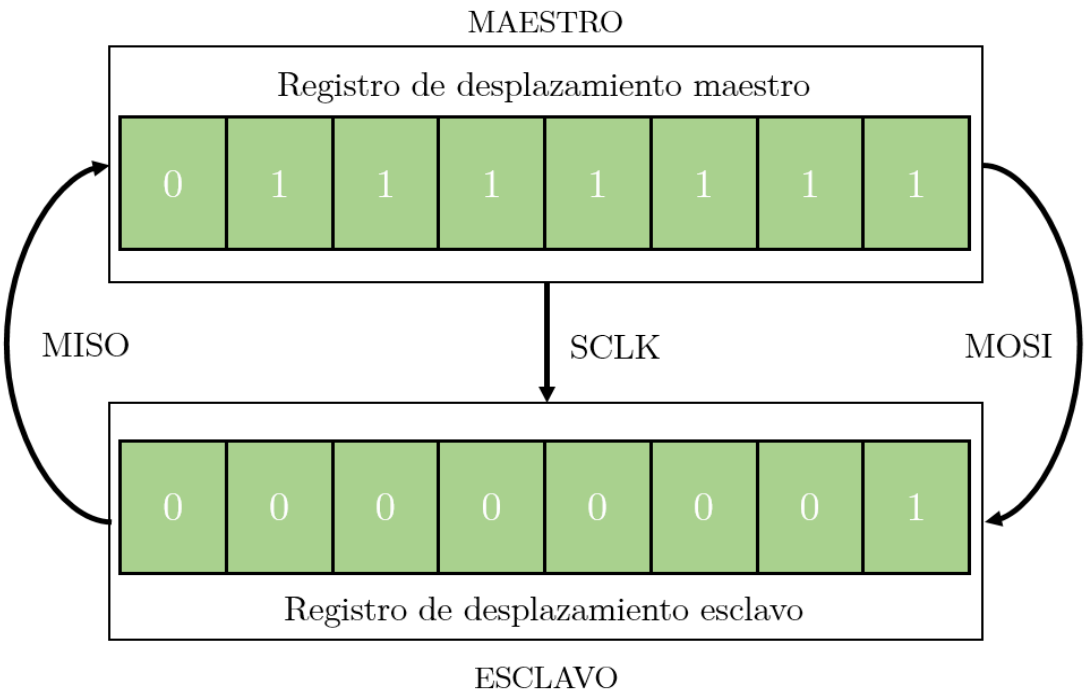

Figura 9: Esquema del funcionamiento de la transmisión de datos por SPI

La señal de reloj puede ser por defecto alta o baja, y la lectura de los datos se puede realizar en el flanco de caída o del reloj o en el de subida, y según la combinación de estos dos factores existen 4 modos de SPI, a los que el maestro tendrá que ceñirse dependiendo del dispositivo esclavo que se conecte.

La elección de esta interfaz respecto a otras es principalmente su velocidad, además de su simple conexión. También la gran cantidad de dispositivos que soportan SPI y la facilidad de la implementación al ser un estándar en la industria.

5.3.2 InfluxDB HTTP API

Antes de explicar el funcionamiento de la API HTTP de InfluxDB hay algunos conceptos previos necesarios para entender la estructura de las bases de datos de Influx.

5.3.2.1 Conceptos básicos

InfluxDB crea una instancia que puede albergar varias bases de datos, las cuales a su vez pueden albergar varias medidas o *measurements*. Los *measurements* son el equivalente a las tablas de las bases de datos relacionales clásicas. Cada *measurement* aloja los valores en columnas, las cuales se pueden declarar como *tags* o *fields*. Cada *measurement* a su vez tiene una marca de tiempo o *timestamp*, la cual si no se especifica es por defecto el tiempo en el instante en que se introducen los datos en la base de datos.

La principal diferencia entre los *tags* y los *fields* es el hecho de que los primeros están indexados, y por lo tanto se pueden hacer peticiones a la base de datos en función de estos. Normalmente los *tags* ayudan a definir el *measurement*, son metadatos, mientras que los *fields* son datos.

Por ejemplo, se podría hacer una base de datos para monitorizar la temperatura de varios ordenadores, en este ejemplo cada measurement podría ser un grupo de ordenadores distinto, mientras que un *tag* identificaría una parte del ordenador y otro el ordenador en concreto, la temperatura sería el *field*. Este ejemplo se muestra en la Tabla 4.

Influx permite añadir varios *fields* y varios *tags* en cada fila de cada *measurement*. Cada uno de ellos forma un *field/tag key*, y su conjunto se denomina *field/tag set*.

Temperatura PCs (*Measurement* #1)			
Parte (*tag* #1)	**ID PC (*tag* #2)**	**Temperatura °C (*field* #1)**	**Tiempo (*Timestamp*)**
GPU	1	33	1575053117
CPU	1	39	1575053117
GPU	2	36	1575053117
CPU	2	42	1575053117

Tabla 4: Estructura de un *measurement* de una base de datos ejemplo

La manera de interactuar con InfluxDB es mediante CLI (Interfaz por línea de comandos, del inglés *Command Line Interface*) o usando su API HTTP.

La primera opción, la CLI, sirve para crear bases de datos y usuarios, cargar configuraciones de InfluxDB, hacer peticiones a la base de datos de forma manual para ver sus contenidos e introducir datos manualmente o desde un archivo. En el presente trabajo se ha usado para hacer pruebas, así como para abrir las bases de datos necesarias. Las demás funciones no serán especialmente útiles ya que utilizaremos Grafana para hacer las peticiones a la base de datos y visualizar la respuesta, y además nos interesa automatizar la entrada de datos.

Por otro lado, la API HTTP nos permite acceder desde el microcontrolador y automatizar la entrada de datos, facilitando la programación.

En caso de que se haga una consulta a la base de datos, la API HTTP de InfluxDB simplemente sirve como método de transporte del lenguaje de peticiones de InfluxDB llamado InfluxQL (del inglés *Influx Query Language*). Este es un lenguaje reminiscente de otros SQL que permite interactuar con InfluxDB. En cambio, si se quieren introducir datos en la base de datos, el cuerpo del mensaje HTTP está compuesto por los datos que queramos introducir, y este ha de seguir la sintaxis establecida por el protocolo de línea de InfluxDB.

En cualquier caso, tanto si es para realizar una consulta como para introducir datos, la petición HTTP también incluye una cadena de consulta o *query string* con varios comandos que la API requiere, como la base de datos a utilizar, o el usuario y la contraseña si se utiliza autentificación. Estos comandos o *query string parameters* son codificados en forma de URL (Localizador de recursos uniforme, del inglés *Uniform Resource Locator*), y varían según el *endpoint* a utilizar.

Los *endpoints* se pueden definir como URLs de una API web donde se espera recibir y desde donde se envía información estructurada. InfluxDB establece principalmente cuatro *endpoints,* en la Tabla 5 se encuentra una breve descripción de estos.

Endpoint	Descripción
/debug/x	Opciones de depuración e identificación de problemas
/ping	Comprobar el estado de InfluxDB
/query	Hacer peticiones a una base de datos
/write	Escribir datos a una base de datos previamente existente

Tabla 5: Breve descripción de los *endpoints* de la API HTTP de InfluxDB

De estos *endpoints*, tan solo se utilizará la API HTTP para el último, que nos permitirá programar el microcontrolador para escribir en la base de datos los datos recogidos por los sensores.

A continuación, se explica el uso de este *endpoint*, empezando por la sintaxis necesaria para la codificación de la URL y acabando con la sintaxis del protocolo de línea que es necesario añadir en el cuerpo del mensaje HTTP.

5.3.2.2 Sintaxis API de influxDB

Los *query strings* compatibles con el *endpoint* /write se muestran en la Tabla 6, traducidos de la referencia de la API de InfluxDB (InfluxData, 2018).
Los *query strings* se introducen tras el *endpoint* separados de este por un signo de interrogación, y entre ellos por un *et* (&), como es típico en la codificación URL. Así, por ejemplo, la petición HTTP para subir una medida a una base de datos llamada *DBtest_tfg*, con el usuario *nil,* la contraseña *nilpw* y una precisión de segundos, con la política de retención de datos por defecto (conservar todos los datos) quedaría como en la Figura 10.

```
POST /write??db=DBtest_tfg&u=nil&p=nilpw&precision=s HTTP/1.1
Host: 192.168.1.37:8086
cache-control: no-cache
[Cabeceras HTTP]

temperatura_PCs,tag_parte=CPU,tag_ID=2 field_temperatura=42
```

Figura 10: Ejemplo petición HTTP a InfluxDB para añadir una medida

Parámetro	Uso	Descripción
db=<database>	Requerido	Base de datos dónde se escribirán los datos especificados en el cuerpo HTTP
p=<password>	Opcional si no hay autentificación	Contraseña
precsion= [ns,u,ms,s,m,h]	Opcional	Precisión de la marca de tiempo, por defecto en nanosegundos y en tiempo Unix
rp= <policy_name>	Opcional	Politica de retención de datos (Por defecto si no se especifica)
u=<username>	Opcional si no hay autentificación	Usuario

Tabla 6: *Query strings* compatibles con el *endpoint* /write de InfluxDB

Como se observa, es necesario utilizar el método POST, especificando en la URI el endpoint y los *query strings*. El host será la dirección IP del ordenador donde se ejecute la instancia de InfluxDB, y el puerto por defecto es el 8086. Siguen las cabeceras HTTP necesarias, y después de la línea en blanco el cuerpo con la medida que vayamos a introducir, que ha de seguir la sintaxis del protocolo de línea, el cual se introduce a continuación.

InfluxDB responde con la respuesta HTTP correspondiente a cada petición, que principalmente nos indicará si la operación ha sido realizada correctamente. Esto se hará mediante los códigos HTTP estándar, principalmente, los códigos HTTP con formato 2xx corresponden a un éxito de la operación, los 4xx a un error y los 5xx a un error interno del servidor.

5.3.2.3 Protocolo de línea de InfluxDB

Como se ha avanzado en el anterior apartado, el protocolo de línea de InfluxDB es el formato de texto con el que se han de incluir los datos a introducir en la base de datos para que la API los entienda correctamente.

Para añadir un solo punto, la estructura básica consta como se explicó de un *measurement* y un *field*, ya que los *tags* no son obligatorios y por defecto InfluxDB añade automáticamente la marca del tiempo al entrar los datos. La sintaxis obliga a poner primero el nombre del *measurement* que se recoge; seguido de este y sin espacios, se introducen los *tags* separados con comas sin espacio si hay varios. En la misma línea y después de un espacio se introducen los *field* separados por comas y sin espacios si hay varios. Finalmente, y si se quiere poner explícitamente una marca de tiempo o *timestamp*, se puede añadir después de otro espacio. La marca de tiempo deberá estar en el formato descrito en (Klyne & Newman, 2002).

Como ejemplo de la estructura descrita, en la Figura 10 se muestra un punto con el *measurement* temperatura_PCs, seguido por dos *tags* y un *field*.

Se pueden introducir varios puntos de datos a la vez en la base de datos añadiendo un salto de línea en el cuerpo del mensaje HTTP después del primer punto introducido.

Todos los campos son cadenas de caracteres, pero los *field* pueden ser también un número entero, un booleano o un real en coma flotante (*float*).

5.3.3 Interacción Grafana-Influx

Conectar Grafana con InfluxDB es un procedimiento sencillo ya que ambas soluciones están preparadas para trabajar entre sí, tan solo hay que proporcionar la localización de la instancia de InfluxDB, usando la dirección ip del *host* y el puerto configurado de InfluxDB, la base de datos a usar, y las credenciales de un usuario de InfluxDB con permisos para realizar consultas a la base de datos, como se muestra en la Figura 11.

Grafana utiliza la API HTTP de InfluxDB usando el *endpoint* /query para solicitar los datos necesarios, lo hace de una forma muy integrada y transparente al usuario,

no se requiere de ninguna programación, tan solo especificar las peticiones necesarias para cada grafico del *dashboard* mediante su GUI (Interfaz gráfica de usuario, del inglés *Graphical User Interface*).

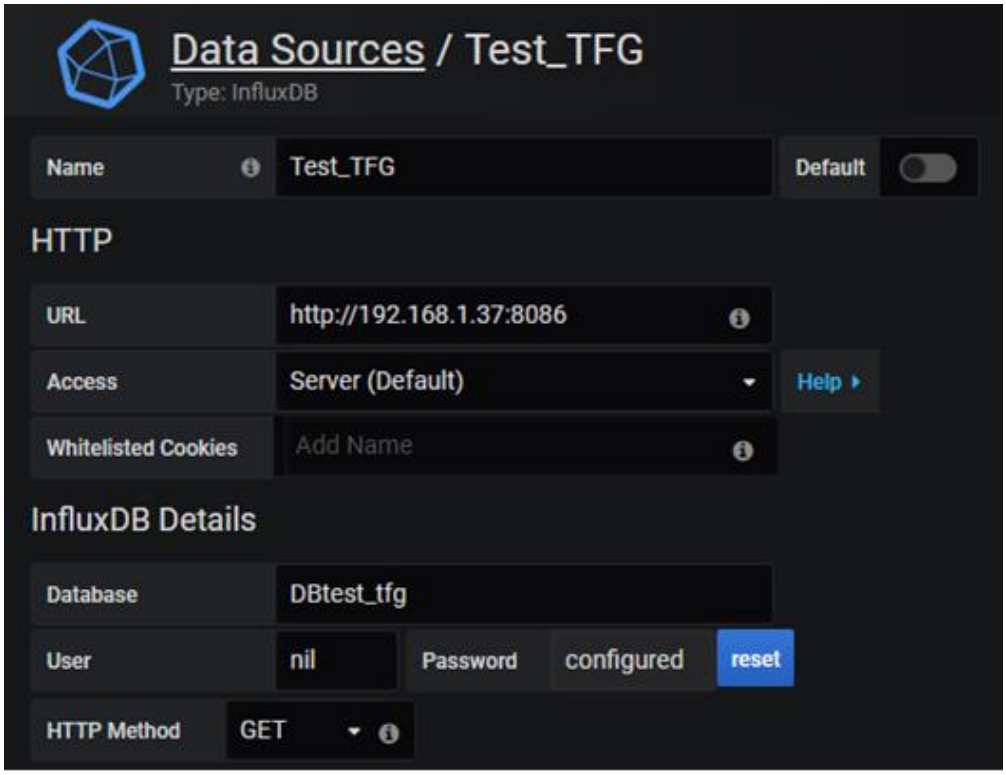

Figura 11: Pantalla de configuración de Grafana para usar InfluxDB

5.3.4 Petición HTTP a Grafana

Grafana proporciona una GUI que permite ver los *dashboard* creados de una manera sencilla. Se puede acceder a la GUI a través de cualquier navegador moderno usando la dirección IP de la máquina donde se aloja el servicio Grafana siendo el puerto por defecto 3000.

5.4 Hardware

El Hardware necesario para el proyecto, a parte del ordenador dónde se ejecuta la base de datos, se compone del microcontrolador, los sensores y los dispositivos de expansión de funcionalidades si fuese necesario. A continuación, se analizan las diferentes opciones del mercado para estos elementos.

5.4.1 Microcontrolador

El requisito principal del microcontrolador es que cuente con capacidades WiFi. Los dispositivos WiFi son aquellos certificados por la WiFi Alliance que cumplen con los protocolos IEEE 802.11 explicados en el capítulo anterior. Así, no se contemplarán dispositivos con otro tipo de conectividad, como radio, infrarrojos o Bluetooth, ya que el uso de WiFi esta más extendido y por lo tanto permite conectar el microcontrolador a una mayor cantidad de dispositivos.

No hay un requerimiento específico de potencia de cálculo del procesador inicialmente, ya que las operaciones a realizar se pueden configurar para usar más o menos recursos. En el capítulo Pruebas y resultados se analizarán las limitaciones técnicas encontradas, ya que a priori solo se pueden estimar a partir de las *datasheet*.

Teniendo en cuenta la disponibilidad en el mercado, se han considerado los dispositivos listados a continuación.

5.4.1.1 Arduino MKR WiFi 1010

Arduino (https://www.arduino.cc/) es una empresa que ofrece un ámplio rango de microcontroladores y accesorios, así como su lenguaje de programación (basado en C) e IDE (entorno de desarrollo integrado, del inglés *Integrated Development Environment*), online y offline. Entre las ventajas de un dispositivo Arduino se encuentra su gran base de usuarios y comunidad activa, así como todos los periféricos probados y/o diseñados para Arduino, entre ellos las placas de expansión o *shield* que añaden funcionalidades complejas.

Arduino ofrece varias opciones para conseguir conectividad WiFi en sus microcontroladores. A cualquier dispositivo Arduino se le puede agregar un *shield* WiFi o un ESP8266. También existen algunos modelos con conectividad nativa, como el MKR WiFi 1010, que incorpora un microcontrolador ESP32 para dar conectividad.

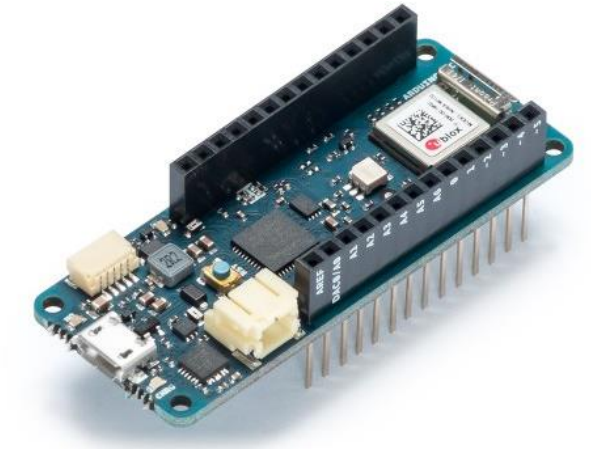

Figura 12: Arduino MKR WiFi 1010

5.4.1.2 ESP8266

Producido por Espressif (https://www.espressif.com/) , el ESP8266 es un modulo de muy bajo precio y tamaño con conectividad WiFi. Se introdujo al mercado en 2014 y con el tiempo ha logrado una gran base de usuarios que han traducido la documentación original china y producido gran cantidad de librerías.

También existe un SDK (Kit de desarrollo de software, del inglés *Sofware Development Kit*) de código abierto que permite programarlo en el IDE de Arduino como cualquier otro Arduino, y por lo tanto lo hace una alternativa muy buena a este, a menor precio y con conectividad WiFi nativa. No obstante, el ESP8266 dispone de menos recursos de Hardware al compararlo con la mayoría de Arduinos, tanto en rendimiento como en entradas y salidas.

El ESP32 es una versión posterior con mejores prestaciones, pero al ser más nuevo no hay tanta información ni librerías disponibles y no se considerará.

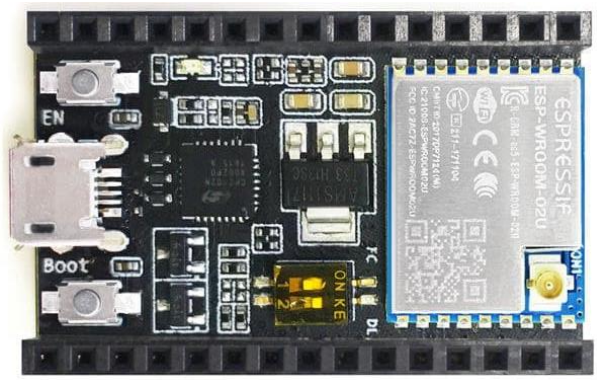

Figura 13: ESP8266

5.4.1.3 Particle Photon

Particle (https://www.particle.io/) es una empresa que ofrece un ecosistema IoT, comprendido por sus microcontroladores, su IDE online y offline, y hasta una plataforma de monitorización de los dispositivos. Este ecosistema permite desarrollar un producto de manera sencilla ya que todas las herramientas están integradas.

El Photon es un dispositivo con conectividad WiFi y tamaño reducido, con unas características técnicas similares al ESP8266, aunque con más entradas y salidas analógicas y a un precio superior.

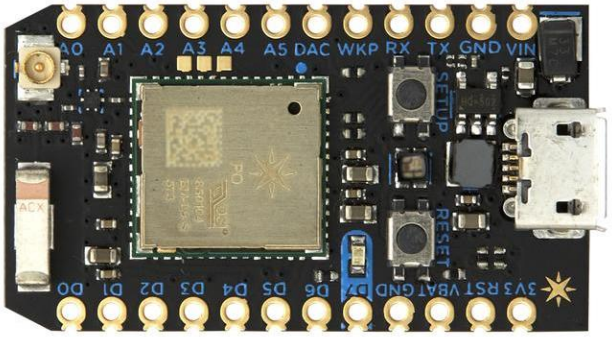

Figura 14: Particle Photon

5.4.1.4 Raspberry Pi Zero W

Raspberry Pi (https://www.raspberrypi.org/) ofrece varias plataformas SBC (ordenador de una sola placa, del inglés *Single Board Computer*). La mayoría son demasiado grandes y potentes, pero la Pi Zero W es pequeña y dispone de conectividad WiFi y Bluetooth. Técnicamente no es un microcontrolador sino un ordenador, por lo que también requiere un sistema operativo y puede no ser tan útil en situaciones donde el control preciso del tiempo es necesario, ya que el sistema operativo puede dar prioridad a otras tareas que no sean el código programado.

Figura 15: Raspberry Pi Zero W

Para comparar las opciones que se han presentado se ha elaborado una tabla comparativa (

Tabla 7) de sus características principales, incluyendo su hardware y precio.

Como se observa, la Raspberry Pi es la alternativa más potente, pero la necesidad de usar una tarjeta MicroSD e instalar un sistema operativo dificulta su uso, además del hecho de que no tiene ninguna entrada analógica y que la potencia de cálculo no es un gran problema para nuestra aplicación, mientras el dispositivo elegido sea capaz de gestionar la red.

En cuanto al Arduino MKR WiFi 1010, no se le encuentra ninguna ventaja en frente a la Particle Photon, siendo esta última más barata y más potente. Además, el microcontrolador de Arduino incorpora un ESP32 para darle conectividad, pero para nuestra aplicación, utilizar el ESP32 o el ESP8266 directamente sería suficiente y ahorraría la placa Arduino.

Dispositivo	Unidad de procesamiento	Velocidad de reloj	RAM	Almacenamiento	Voltaje interno	Pines analógicos	Pines digitales	Coste (aprox.)
Arduino MKR 1010	SAMD21 Cortex + ARM MCU	48MHz	32KB	256KB	3,3V	7ADC 1 DAC	8	21€
ESP8266	Tensilica L106	80-160MHz	50KB	2MB SPI	3,3V	1 ADC	11	1-2€
Particle Photon	STM32 ARM Cortex M3	120MHz	128KB	1MB	3,3V	8 ADC 2 DAC	18	17,2€
Raspberry Pi Zero W	ARM11 Broadcom	1 GHz	512MB	MicroSD	3,3V	0	28	10,5€

Tabla 7: Comparación de las características de las distintas alternativas de microcontroladores, información proporcionada por los fabricantes

Esto deja como opciones la Particle Photon y el ESP8266. La principal ventaja de la Particle Photon radica en que ya dispone de pines analógicos, pero en cuanto a las otras características no distan mucho entre ellos, siendo más interesante por precio usar el ESP8266 con un ADC adicional que proporcione más entradas analógicas.

En resumen, se ha visto como el microcontrolador ESP8266 es más económico que la competencia, permite un diseño del prototipo más simple al no tener que añadir *shields* como en otros modelos de Arduino y dispone de una gran base de usuarios que han creado multitud de librerías, por lo que es una opción ideal para este proyecto.

Finalmente, hay que destacar que existen otras opciones, pero sea por su baja disponibilidad en el mercado, o por que al integrar funciones innecesarias para este proyecto su precio es demasiado alto, han sido descartadas. Por ejemplo, la familia ARTIK de Samsung o los Intel Edison/Galileo.

5.4.2 Conversor analógico digital

Como se ha comentado, la necesidad de tener varias entradas analógicas para poder utilizar varios sensores analógicos a la vez requiere el uso de un ADC adicional.

Para escogerlo, se valorarán principalmente tres aspectos: su precio, sus características y su uso anterior en otros proyectos. Si el ADC se ha usado previamente, probablemente haya alguna librería de código que se pueda usar para aprovechar desarrollos anteriores.

Se buscará un ADC que se comunique con el ESP8266 mediante SPI, aprovechando que este tiene la capacidad de usar esta interfaz, para lograr una transmisión de datos más rápida. De cara a futuras aplicaciones también se tendrá en cuenta de que disponga de 8 canales.

Tras filtrar varios ADC comerciales con estas características añadiendo la restricción de que la tensión de alimentación sea de los 3,3V que ofrece el ESP8266, se han obtenido los siguientes modelos, mostrados en la Tabla 8.

Modelo	Resolución	Frecuencia de muestreo	Arquitectura	Precio
ADC120 STMicroelectronics	12bit	1 MS/s	SAR	4,28€
ADS1118 Texas Instruments	16bit	860 S/s	Sigma-Delta	4,96
MCP3008 Microchip	10bit	200 kS/s	SAR	1,97€
MCP3208 Microchip	12bit	100 kS/s	SAR	3,13€

Tabla 8: Comparación de los modelos ADC que cumplen los requisitos establecidos

Viendo las características, podemos descartar el MCP3008 ya que se considera una resolución demasiado baja. Para el ADC120 de STMicroelectronics no se ha encontrado ninguna librería que lo implemente, mientras que para el ADS1118 y el MCP3208 sí se han encontrado, aunque la del ADS1118 podría presentar problemas de compatibilidad ya que la librería encontrada solo es para Arduino.

Aunque el modelo ADS1118 de Texas Instruments presenta una muy buena resolución, su frecuencia de muestreo es significativamente inferior a la alternativa de Microchip. Dependiendo de la aplicación esto nos puede limitar, por lo que se ha decidido inclinarse por el ADC MCP3208 de Microchip, el cual presenta también un coste menor de adquisición.

5.4.3 Sensores

Para realizar las pruebas de funcionalidad, se ha decido usar un sensor de corriente y un sensor de CO_2.

5.4.3.1 Sensor de corriente

Con el sensor de corriente se podrá realizar una estimación del consumo del sistema HVAC o de cualquier electrodoméstico. Adicionalmente, recogiendo suficientes muestras, permite evaluar la calidad de la red eléctrica a través de los armónicos y distorsión armónica total que esta presenta. Estas características se obtendrán al hacer un procesado de las muestras dentro del microcontrolador, usando una FFT (transformada de Fourier rápida, del inglés *Fast Fourier Transform*), ya que pasar todas las muestras a la base de datos no es práctico.

La elección del sensor de corriente se ha hecho teniendo en cuenta las alternativas en el mercado: sensores de corriente invasivos y no invasivos. Se ha optado por estos últimos, en concreto por el sensor SCT-013 porque es del tipo no invasivo, tiene un precio reducido de unos 3-4€ y ha sido ampliamente utilizado. Un ejemplo de sensor invasivo sería el Sodial-ACS712. Adicionalmente, se disponía de él en el laboratorio al iniciar el trabajo.

El sensor no invasivo SCT-013 funciona como un transformador: cuenta con un bobinado que se cierra alrededor de un cable por el que pasa una corriente alterna, de este modo el cable le induce una señal eléctrica, con una amplitud menor según número de espiras del bobinado, que varia entre modelos.

La mayoría de modelos cuentan con una resistencia de carga para convertir esta corriente inducida en un voltaje cuantificable por el ADC.

Figura 16: Sensor de corriente no invasivo SCT-013

5.4.3.2 Sensor de CO_2

En cuanto al sensor de CO_2, éste permite realizar estimaciones de la calidad del aire de una habitación, además de poder inferir la ocupación de una sala. Para esta última operación se requiere de un sensor suficientemente preciso o conformarse con obtener resultados solo en el caso de que la sala presente una alta ocupación o durante un tiempo extendido y sin ventilación.

Se han considerado varios sensores para llevar a cabo esta tarea, entre ellos dos sensores de funcionamiento químico, los cuales detectan la concentración de CO_2 según una reacción química y suelen ser más baratos que sus contrapartidas de funcionamiento mediante un sensor no dispersor infrarrojo o NDIR (del inglés *Non Dispersive Infrared*).

En total se han comparado 6 sensores evaluando sus características técnicas de medida de la concentración del gas, dejando a parte otras características de tamaño o conectividad. Los resultados de la búsqueda de información se muestran en la Tabla 9.

Sensor	Tipo	Rango [ppm]	Exactitud [ppm]	Precio [€]
MQ-135	Químico	10-10.000	±200*	3,40
MG811	Químico	400-10.000	±200 *	18,10
MH-Z16	NDIR	400-10.000	±200	28,96
K30	NDIR	0-5.000	±30	75,85
CozIR	NDIR	0-10.000	±50	97,27
ExplorIR	NDIR	0-1.000.000	±70	97,30

Tabla 9: Datos de los fabricantes para varios sensores de CO_2 *Según calibración

Los sensores MQ-135, MG811 y MH-Z16 presentan un precio más asequible, y sus características son similares. Si se requiriera una precisión superior se podría optar por el sensor K30 ya que ofrece un buen rango para detectar la calidad del aire y una muy buena exactitud. Entre los tres anteriores, el sensor MQ-135 es el que presenta mayor facilidad de compra.

En conclusión, debido a que los sensores de CO_2 más precisos tienen un precio prohibitivo, y al ser su uso en el presente trabajo tan solo demostrativo, se ha optado por un sensor menos preciso con un precio más reducido como es el MQ-135.

El MQ-135 es un sensor que dispone tanto de salida digital como analógica, con un precio que oscila en torno a los 1-2 €. El funcionamiento de este sensor está basado en el aumento de conductividad por parte del SnO_2 (dióxido de estaño) al entrar en contacto con gas CO_2.

Figura 17: Sensor de gas analógico-digital CO_2 MQ-135

Los *datasheet* del sensor SCT-013, el MQ-135 así como del ADC y el microcontrolador se pueden encontrar en el Anexo C, D, E y F respectivamente, donde se muestran sus características más importantes así como su funcionamiento interno e interfaces detalladamente.

5.5 Software

En este apartado se discutirá el diseño del programa ejecutado en el microcontrolador, el objetivo del cual es leer los sensores, realizar la FFT y enviar los resultados a la base de datos. Posteriormente se comentará la configuración de InfluxDB y Grafana.

5.5.1 Programación del microcontrolador

La programación del microcontrolador se puede dividir en varias tareas, como se muestra en el diagrama de flujo de la Figura 18. Se distingue la parte de definición de constantes, la adquisición de los datos y el procesado de estos para poder enviarlos a través de la API de InfluxDB correctamente. Por otro lado, la parte de comunicación se ejecuta paralelamente.
A continuación se comenta la arquitectura general del programa.

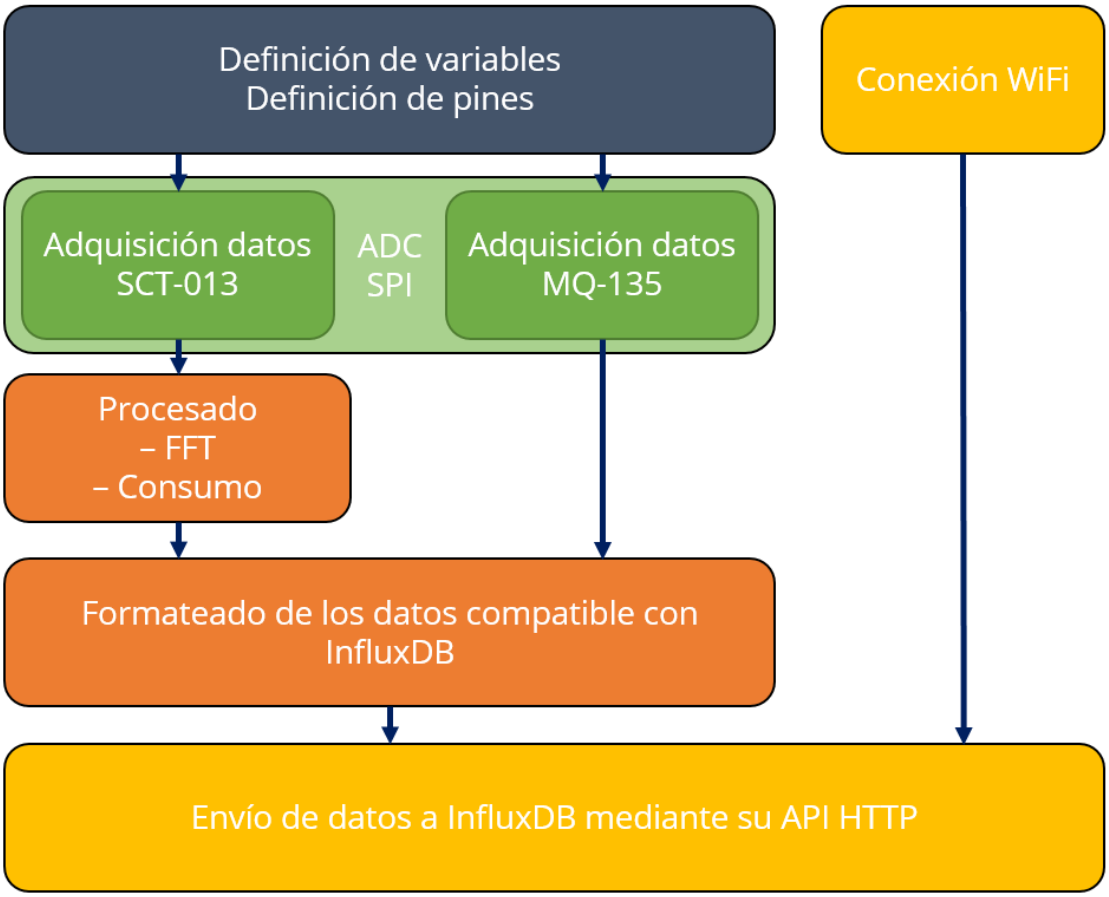

Figura 18: Diagrama de flujo del programa

En la primera parte del código se incluyen varias librerías que se utilizarán durante el resto del programa: para el ADC, para hacer la FFT, y las necesarias para establecer la conexión WiFi y hacer peticiones HTTP a InfluxDB (Figura 19).

```
#include <ESP8266WiFi.h>                    //Librerias WiFi
#include <ESP8266WiFiMulti.h>
#include <InfluxDb.h>                       //Libreria InfluxDB
#include <arduinoFFT.h>                     //Libreria FFT
#include <SPI.h>                            //Libreria SPI para el ADC
#include <Mcp320x.h>                        //Libreria ADC
```

Figura 19: Librerías utilizadas para el código

A continuación, se establecen ciertas constantes necesarias como los pines SPI, las credenciales de redes WiFi a utilizar, la base de datos de InfluxDB y sus credenciales, y otras variables y objetos requeridos.

Siguiendo a la inclusión de las librerías y la declaración de variables, se procede a la inicialización del WiFi, la conexión a la base de datos y la inicialización del SPI mediante la función *setup*().

Dentro del bucle principal, y si se cumple la condición de estar conectado a la red WiFi, se ejecutan las operaciones principales.

El bucle principal *loop()* empieza con la adquisición de los datos de los sensores utilizando la librería del ADC. Por un lado se realiza la adquisición de los datos del sensor de gas CO_2 (Figura 20) y por otro el muestreo de la señal eléctrica recogida por el sensor SCT-013 (Figura 21).

```
double concentracionco2(){//Función para obtener la concentracion de CO2
  float ratio=obtenerRS()/R0;
  return regrA * pow(ratio, -regrB);
}
float obtenerRS(){                  //Obtencion de la resistencia del sensor
int raw=adc.read(MCP3208::Channel::SINGLE_0);    // Correspondiente a la
return ((4095/(float)raw)-1)*RL;            //concentracion actual de co2
}
```

Figura 20: Cálculo de la concentración de CO_2

```
float MuestreadoyCorriente(){//Funcion para el muestreado de la señal
  float corriente;           //y calculo de la corriente RMS
  float sum =0;
  microseconds = micros();
  for(int i=0; i<samples; i++)                          //Muestreado
  {
      vReal[i] = adc.read(MCP3208::Channel::SINGLE_7);
      vImag[i] = 0;
      while(micros() - microseconds < sampling_period_us){//Loop espera
      }
      microseconds += sampling_period_us;
  }
  for(int i=0; i<samples; i++)
  {
    vReal[i] = (vReal[i]+4) * ADCV / 4095.0;//Voltaje calibrado ADC
    vReal[i] = fmap(vReal[i], VMIN, VMAX, -FACTOR, FACTOR);//Mapeado
    sum += sq(vReal[i]);
   }
   corriente=sqrt(sum/samples);                        //Calculo RMS
   return(corriente);
  }
```

Figura 21: Muestreado y cálculo de la corriente RMS

La FFT y la corriente RMS (valor cuadrático medio, del inglés *Root Mean Square*) se obtienen a partir de este muestreado de datos que proporciona el SCT-013, como se muestra en la Figura 21 y en la Figura 22 respectivamente.

El consumo aproximado se obtiene multiplicando en el post-proceso el valor de la corriente por una constante que se puede calibrar (230V RMS para una red doméstica en España). De la FFT se obtiene la frecuencia principal de la red, así como los intervalos de frecuencias presentes en la red con su amplitud correspondiente. Según el ruido de la señal, la frecuencia de muestreo y el número de muestras, estos intervalos serán mayores o menores y la magnitud de la amplitud más realista o no.

```
double FFTexe(){
  FFT.Windowing(vReal, samples, FFT_WIN_TYP_HAMMING, FFT_FORWARD);//Ponderar ventana
  FFT.Compute(vReal, vImag, samples, FFT_FORWARD);              //Ejecutar FFT
  FFT.ComplexToMagnitude(vReal, vImag, samples);             //Computar magnitudes
  double x = FFT.MajorPeak(vReal, samples, samplingFrequency);//Frecuencia dominante
  return(x);
}
```

Figura 22: Computación de la FFT

Los resultados de las operaciones descritas anteriormente se tratan en funciones auxiliares dentro del bucle principal que se encargan de formatear los valores recogidos en forma de *measurements*, *tags* y *fields* y guardarlos en un objeto. Finalmente, la función *influx.write()* es la encargada de escribir todos los datos

formateados a la vez, lo cual mejora notablemente el rendimiento del programa al solo requerir una conexión por bucle. Después de un retraso modificable, se repite el bucle principal.

Este proceso se muestra en la Figura 23.

```
void loop() {
if ((WiFiMulti.run() == WL_CONNECTED)) {    //Comprobar la conexion

      float CorrienteRMS = MuestreadoyCorriente();

      yield();

      InfluxData m1 = consumo(CorrienteRMS);//Preparación datos:
      influx.prepare(m1);                   //corriente consumida
      yield();

      double Frec_dom= FFTexe();//Ejecuta el FFT y retorna
      yield();                  //la frecuencia dominante
      float valorTHD = calculoTHD();

      yield();
    for(int i=0; i<(samples/2); i++){//Preparación datos a enviar de FFT
       InfluxData m = FFTdata(i);
       influx.prepare(m);
       yield();
       }

       InfluxData frecthd = frec_y_thd(Frec_dom, valorTHD);//Preparacion datos:
       influx.prepare(frecthd);                        //frec. dominante y THD

        double ppmco2=concentracionco2();   //Obtener concentracion CO2
        InfluxData co2 = co2formato(ppmco2);//Preparacion datos a enviar CO2
        influx.prepare(co2);

      influx.write();// Enviar todos los datos preparados y
      yield();       //ceder paso a procesos secundarios
 }
else {                                  //En caso de conexión fallida avisar
   Serial.printf("No es posible la conexion WiFi\n");
   delay(5000);                         //y esperar antes de reintentar
    }

  delay(2000); //Repetir cada 2 segundos mas el tiempo de ejecución
}
```

Figura 23: Bucle principal del programa

Hay una serie de funciones extra que se encargan de hacer operaciones secundarias, como los cálculos de la FFT, la lectura de vectores o funciones matemáticas de mapeado de los voltajes. Debido a que el microcontrolador no acepta valores de entrada negativos, se requiere el mapeado para su correcto funcionamiento. La señal se desplaza mediante un divisor de voltaje para que lleguen solo valores positivos al microcontrolador. Para obtener los valores reales

(tanto positivos como negativos) para el resultado final, hay que transformar mediante el código la señal modificada y mapearla a sus valores originales.

Cabe mencionar que durante todo el programa es necesario ceder el paso a procesos que se ejecutan en segundo plano mediante la función *yield().* Esto es así debido a que el microcontrolador dispone de recursos limitados y algunas operaciones pueden tardar demasiado tiempo en realizarse. Si no se cede el paso entre operaciones contiguas, los procesos en segundo plano como la conexión WiFi no se ejecutan correctamente.

El microcontrolador dispone de varios WDT (Temporizadores perrito-guardián, del inglés *Watch Dog Timers*), habilitados tanto por software como por hardware, que resetean la placa si el tiempo de ejecución de un bucle supera cierto tiempo. Esto es una medida de seguridad para evitar que la placa se quede atascada durante una operación y haya que resetearla manualmente.

Los mecanismos incluidos en la programación para recuperar el microcontrolador en caso de error, así como los instantes en que se da paso a los procesos en segundo plano se muestran detalladamente en el apartado 7.1, Programación a prueba de errores.

El código completo del programa escrito en C++ se encuentra en el Anexo A.

5.5.1.1 Adquisición de datos y procesado

Se distinguen tres casos según el sensor utilizado y los datos a adquirir.

5.5.1.1.1 Adquisición de datos para la FFT

Para poder hacer la FFT, y siguiendo el teorema de muestreo de Nyquist-Shannon, se escogerá una tasa de muestreo de como mínimo el doble de la frecuencia del armónico a estudiar. Este teorema demuestra que se puede reconstruir una señal periódica contínua a partir de un muestreado de esta de al menos una frecuencia del doble de la máxima presente en la señal a reconstruir.

Teniendo en cuenta las características del ADC y el tiempo de cálculo requerido para otras funciones del programa, que deben poderse realizar a tiempo, la frecuencia escogida finalmente es de $F_{muestreo} = 640\,\text{Hz}$. Se ha elegido un muestreo de $N_{muestras} = 128$, que es el máximo al que se ha podido llegar con el

ESP8266 por limitaciones de memoria. Esto permite estudiar hasta el 6º armónico, el cual, teniendo en cuenta la frecuencia de la red española de 50 Hz ha de estar situado en los 300 Hz que es menos de la mitad de los 640 Hz con que se muestrea, por lo tanto, se satisfacen los criterios del teorema de muestreo de Nyquist-Shannon.

La combinación elegida de la tasa de muestreo y el número de muestras con que se ejecutará la FFT nos permitirá discretizar el espacio de frecuencias de la señal en: $\frac{N_{muestras}}{2} = \frac{128}{2} = 64$ bandas o *bins* de $\frac{F_{muestreo}}{N_{muestras}} = \frac{640}{128} = 5\,Hz$ cada una, a las cuales se les asigna su amplitud de señal correspondiente.

La explicación de la transformada discreta de Fourier y de la FFT queda fuera del alcance del trabajo, pero a grandes rasgos, a la señal muestreada se le aplica una ventana: una serie de coeficientes que multiplican la señal para forzar los extremos de la secuencia de valores de la muestra a un valor nulo.

Aplicar la ventana permite evitar la discontinuidad que se crea en los valores extremos de cada lado de la secuencia de muestras recogidas. La secuencia finita de muestras de la señal se interpreta como un múltiple entero de un periodo de una señal periódica no finita, cuando realmente es un múltiple no entero, y, por lo tanto, los extremos al repetirse son discontinuos. Sin una ventana, esta discontinuidad causaría la aparición de falsas frecuencias en el análisis de la señal. Existen varias ventanas, pero se ha aplicado la ventana Hamming, de uso general (National Instruments, 2016).

Después de aplicar la ventana, se computa la FFT con lo que se obtienen dos vectores, uno con los valores reales y el otro con los imaginarios de un vector complejo. Cada vector tiene la mitad de longitud del vector de muestras, siendo cada posición del vector una *bin* de frecuencias. Finalmente, de estos vectores se extrae la magnitud y la fase de las señales que componen nuestra señal original.

5.5.1.1.2 Adquisición de datos para el consumo de corriente

Para obtener el consumo estimado se requiere un valor de corriente, proporcionado por el sensor SCT-013, así como de una tensión estimada a unos 230 V.

La corriente se consigue realizando un muestreo de la señal del sensor, de este muestreo se hace la media cuadrática de las muestras para obtener el valor eficaz de la corriente que pasa por el conductor dónde tengamos el sensor posicionado.

Previamente se ha de transformar el voltaje medido por el sensor al rango de valores necesario (el sensor produce valores de 0 a 1 voltios RMS o de unos 1,41 V pico a pico, mientras que el ADC está alimentado por una tensión de 3.3 V). Esto implica que la señal de entrada llega de -1,41 V a 1,41 V al ADC. Como el ADC no es capaz de medir valores negativos, se ha decidido implementar un desplazamiento de la señal en $\frac{3{,}3\ V}{2} = 1{,}65\ V$ mediante un divisor de voltaje con un condensador que permite el paso de la corriente alterna.

De este modo, la señal llega al ADC de $1{,}65\ V - 1{,}41\ V = 0{,}24\ V$ a $1{,}65\ V + 1{,}41\ V = 3{,}06\ V$ lo cual está dentro del rango de valores del ADC, que puede captar señales desde 0 V a la corriente de alimentación de 3,3 V. Dentro del programa se hace el proceso inverso de mapeado para trabajar con la onda sinusoidal real.

La frecuencia de muestreo ($F_{muestreo}$) necesaria para computar la señal de corriente desde el sensor SCT-013 no requiere ser tan alta como la necesaria para estudiar los armónicos. Igualmente, para lograr una mayor exactitud y no repetir la operación de muestreo de la señal, esta solo se realiza una vez y se usa tanto para la FFT como para encontrar la corriente que pasa por el hilo conductor.

5.5.1.1.3 Adquisición de datos para el sensor de CO_2

Para el sensor de CO_2, sólo es necesario transformar la señal de éste a la variable física de concentración del gas y publicar esta a intervalos regulares en la base de datos junto con los demás datos.

El sensor MQ-135, está físicamente compuesto por un divisor de tensión: una resistencia conocida (RL) de 1 kΩ y la resistencia del sensor (RS). Para encontrar la concentración real del gas es necesario conocer en todo momento el valor de ambas resistencias, la curva que relaciona su ratio con la concentración de CO_2 y la resistencia del sensor en condiciones conocidas (R0).
El microcontrolador se encarga de calcular RS a partir de los valores leídos por el ADC, la curva se obtiene del *datasheet* incluido en el anexo X y R0 se estima según una concentración media de 400 ppm en la ciudad de Barcelona.

De la curva del *datasheet* podemos extraer los valores correspondientes al CO_2 (Figura 24). A partir de los valores extraídos, y dado que la variación de concentración en función de RS/R0 es prácticamente lineal en el gráfico doble-logarítmico, será conveniente realizar una regresión con una ley potencial, como se ve en la Figura 25.

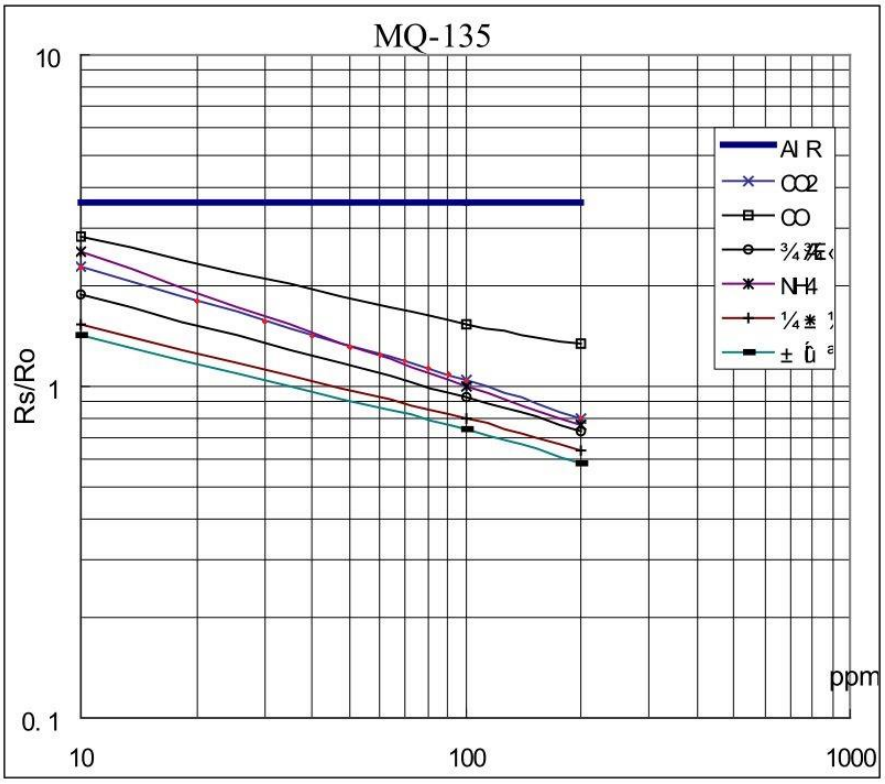

Figura 24: Gráfico doble-logarítmico extraído del *datasheet* del sensor MQ-135 expresando la variación de ratio de resistencias en función de la concentración en ppm de varios gases

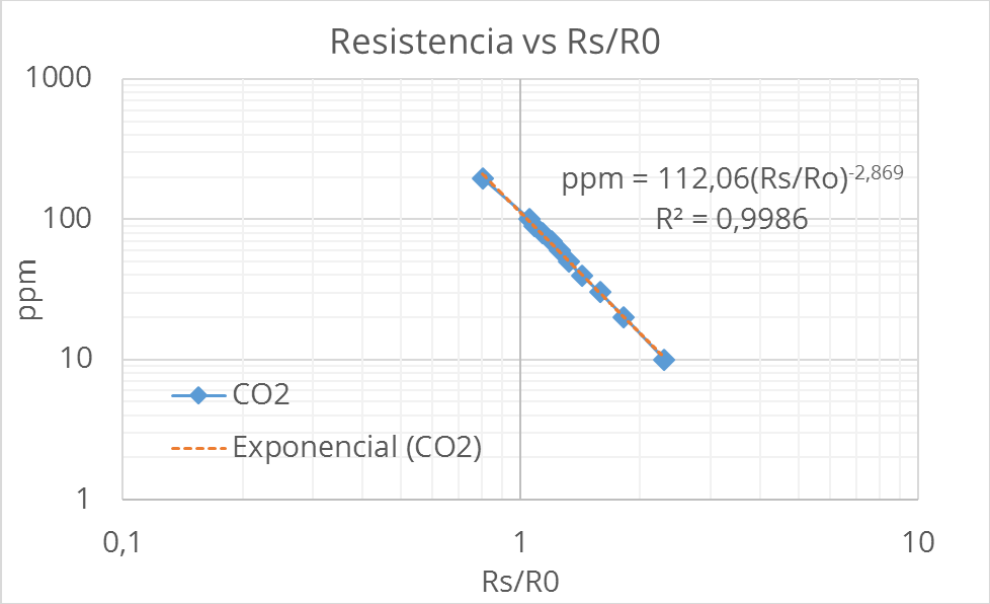

Figura 25: Regresión extraída del gráfico proporcionado por el *datasheet* del sensor MQ-135. Nótese que se expresa la concentración en ppm en función de la ratio de resistencias

Para el valor R0, se mantiene el sensor en una concentración conocida de CO_2, se calcula el valor de RS y se despeja R0 de la ley potencial:

$$ppm = 112\left(\frac{RS}{R0}\right)^{-2,87} \rightarrow$$

$$R0 = RS \cdot \left(\frac{ppm}{112}\right)^{\frac{1}{2,87}}$$

5.5.1.2 Introducción de datos a InfluxDB

Como se ha avanzado anteriormente, se envían las tres señales a la vez dentro del mismo mensaje HTTP a InfluxDB: el resultado de la FFT, los datos del consumo a partir de la corriente registrada por el sensor SCT y la concentración de gas CO_2.

Se formatean los datos según el protocolo de línea añadiendo los *measurements*, *tags* y *fields* y se envían finalmente todos juntos para ahorrar el procesamiento requerido para preparar el mensaje HTTP. Este paso es importante ya que, de lo contrario, no se podrían enviar los resultados de la FFT, debido a que enviar las 64 *bins* requeriría 64 mensajes, siendo el tiempo de preparación del mensaje muy elevado, y retrasando el cálculo y el envío de los demás datos sustancialmente.

No obstante, esta metodología crea un problema: debido al tratamiento de los datos de InfluxDB, dos medidas con los mismos *tags* que lleguen a la vez a la base de datos (mismo *timestamp*) se consideran iguales, sin importar el valor de los *fields*. Para solucionar esto se han asignado como *tags* y como *fields* los valores de la banda de frecuencia enviados de la FFT, creando unicidad para cada medida. Además, para que se ordenen correctamente, se ha debido sumar 1000 a las medidas de frecuencia, de modo que 200 Hz se almacenan como 1200 Hz y se ordenan detrás de 50 Hz que se almacenan como 1050 Hz.

5.5.2 Configuración de InfluxDB

La base de datos tiene que estar configurada para aceptar las tres señales. Esto se consigue estableciendo tres *measurements*: uno para los resultados de la FFT, otro para los datos de consumo y un tercero para la concentración de CO_2. La base de datos dónde se almacenan los datos debe haber sido creada con anterioridad.

En la figura Figura 26 se muestra esquemáticamente la estructura final de los datos almacenados en InfluxDB. Como se ha descrito en los capítulos anteriores, cada punto de datos de la base de datos, está compuesto por un *measurement*, con sus *tags* y sus *fields* más un *timestamp* que le asigna el servidor al recoger la información.

Todos los *tags* son obligatoriamente cadenas de caracteres, mientras que los demás valores se trabajan como *floats* o *double* en caso de las frecuencias.

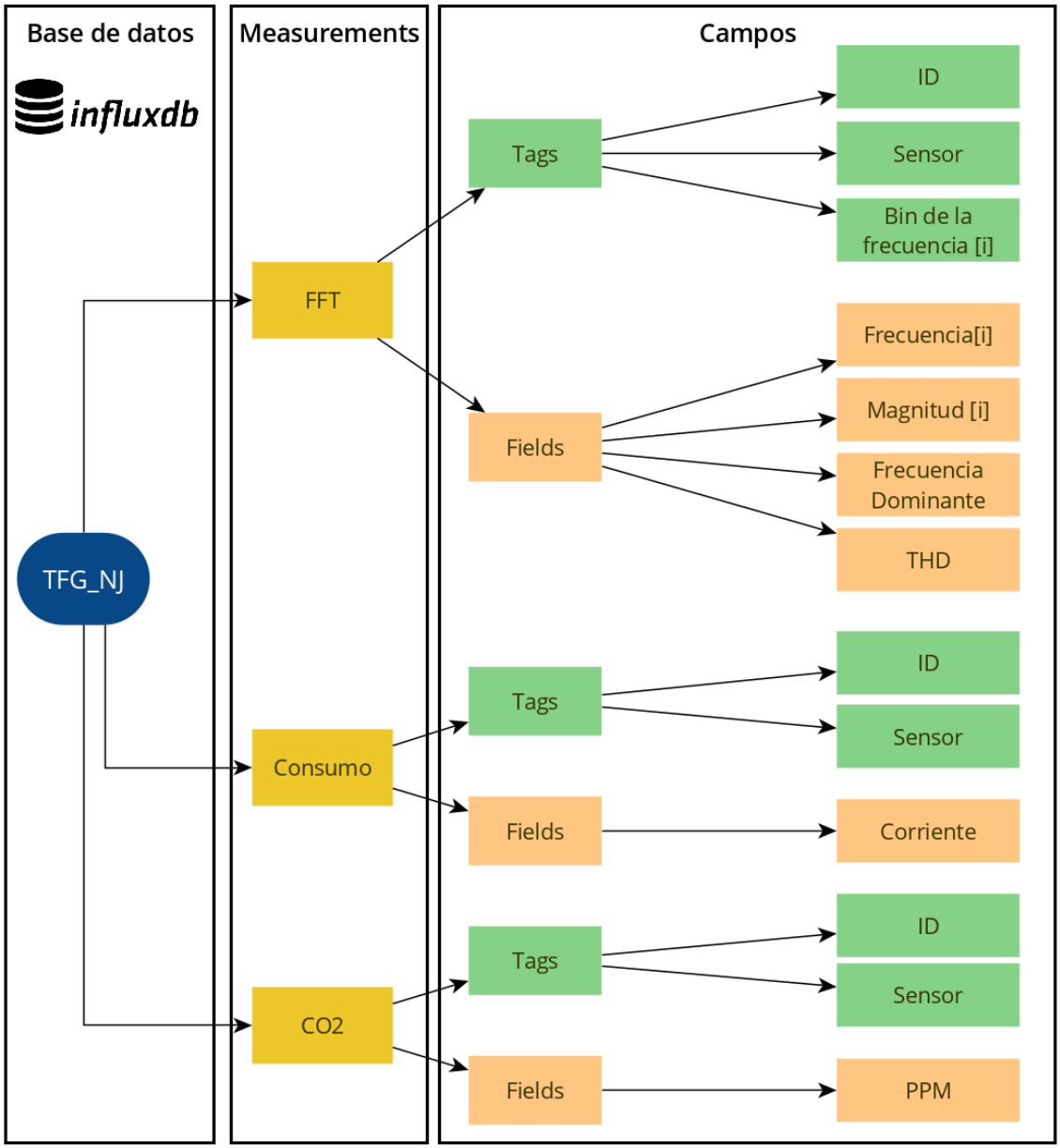

Figura 26: Estructura de almacenamiento de la base de datos creada en InfluxDB

Se ha contemplado en todo momento mantener la base de datos segura, por lo que se ha activado la opción de exigir autentificación de InfluxDB, por lo que se han añadido las credenciales en la programación del ESP8266. En un futuro también sería interesante contemplar el uso del protocolo HTTPS en vez de HTTP, pero este requiere de un certificado TLS (seguridad en la capa de transporte, del inglés *Transport Layer Security*) del que no se dispone, así como una configuración más extensa, por lo que en el presente trabajo se ha omitido este paso.

5.5.3 Configuración Grafana

En cuanto a la configuración de Grafana, se ha creado un *dashboard* que muestra los *measurements* recogidos por InfluxDB en varios paneles interactivos. A modo de ejemplo, en la Figura 27 se muestra como quedaría un *dashboard* finalizado.

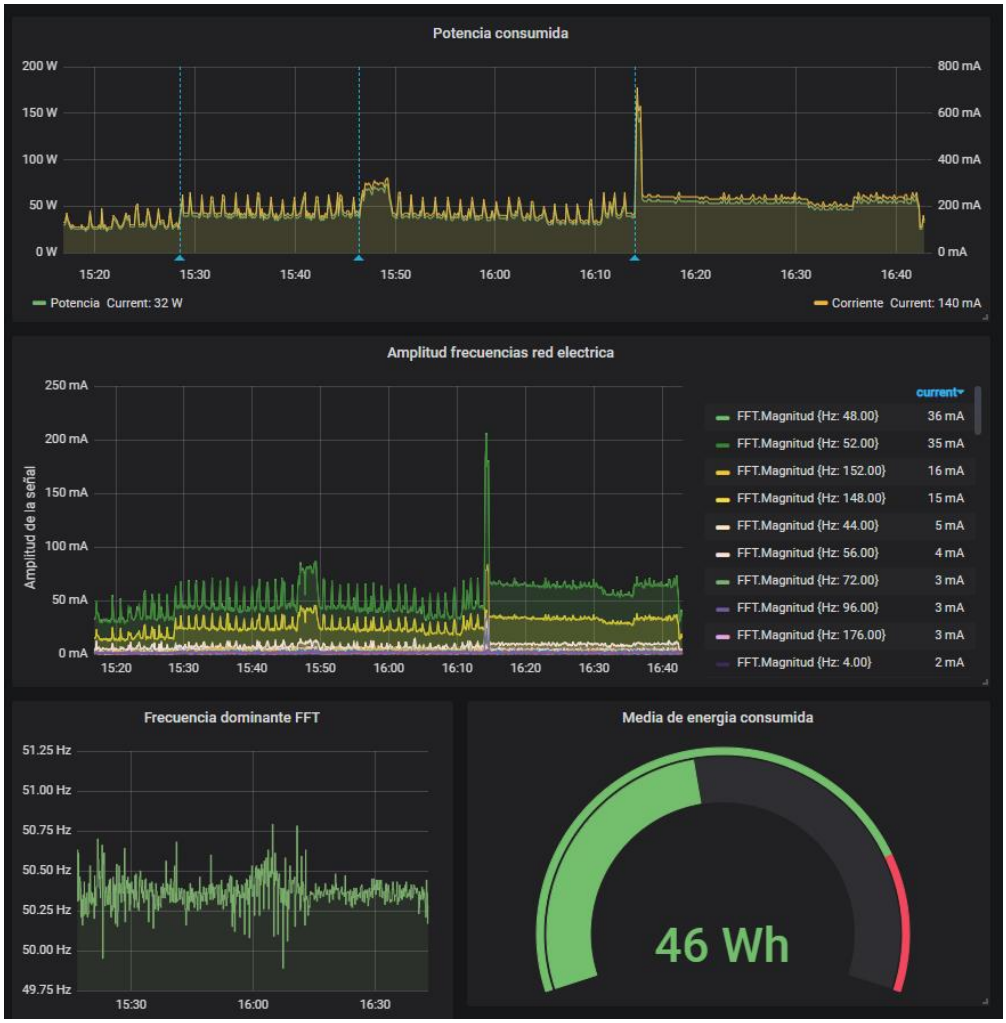

Figura 27: Vista general de un d*ashboard* en Grafana con los datos de consumo y FFT

En cada panel se requiere especificar una petición para una base de datos en concreto, y se elije la visualización preferida por el usuario. Por ejemplo, como se muestra en la Figura 28, para mostrar la potencia consumida, se escoge un gráfico de líneas solicitando a la base de datos los valores de corriente multiplicados por una constante para obtener la potencia consumida.

Figura 28: Petición a la base de datos para la visualización de un panel

En el *dashboard* final, se ha escogido representar en paneles los siguientes datos:

- La potencia consumida en función del tiempo, calculada a partir de la corriente suministrada por el cable que se monitorea. Se muestran ambas en forma de gráfico. Ejemplo de la Figura 28.
- Los armónicos y demás frecuencias presentes en la señal. Se muestran de diferentes formas: en forma de gráfico temporal, en forma de espectro de frecuencias y en forma de tabla, lo que permite extraer los valores fácilmente además de visualizarlos de una forma intuitiva (Figura 29).

Figura 29: Visualizaciones de la FFT

- La distorsión armónica total, cálculo realizado en el microcontrolador. También se muestra en forma de gráfico temporal, lo que permite ver su evolución y cambios comparándola con la potencia consumida.

  En la Figura 30 se muestra la variación de la distorsión armónica total al tener conectado un ordenador y un cargador de móvil al circuito monitorizado, desconectar el cargador de móvil y, al cabo de unos minutos conectarlo otra vez. Esta información se podría usar para identificar dispositivos conectados a la red.

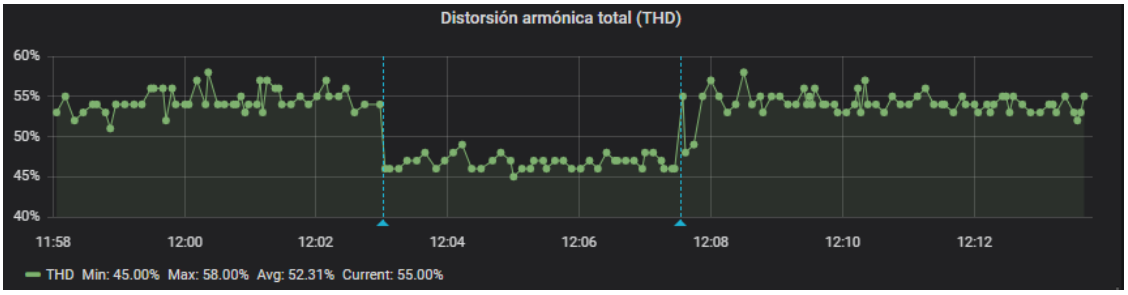

Figura 30: Variación de la distorsión armónica total según las cargas conectadas

- La energía consumida, que se calcula en Grafana promediando los valores anteriores de potencia consumida. Por ejemplo, Figura 27.
- La evolución de la frecuencia dominante, obtenida en el microcontrolador en función de los datos obtenidos de la FFT. Su valor corresponde a la interpolación del valor de frecuencia con mayor magnitud. Se muestra en forma de gráfico temporal (Figura 31).

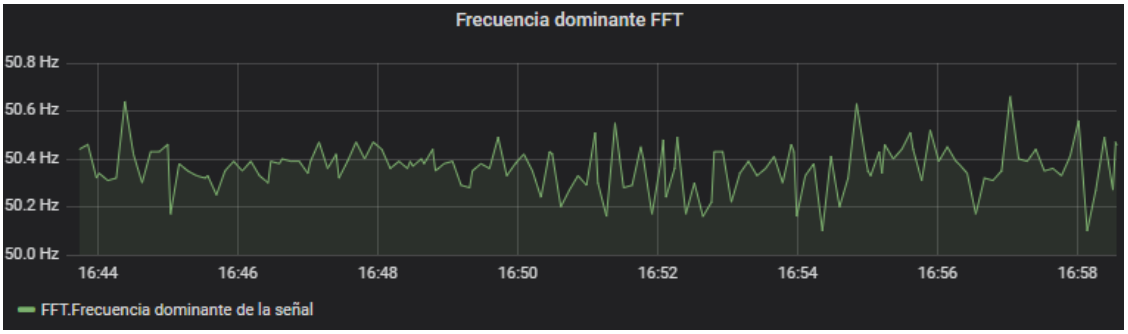

Figura 31: Frecuencia dominante de la red monitorizada

- La concentración de gas CO_2 en el lugar monitorizado, en forma de indicador que muestra varios límites y el valor actual (Figura 32).

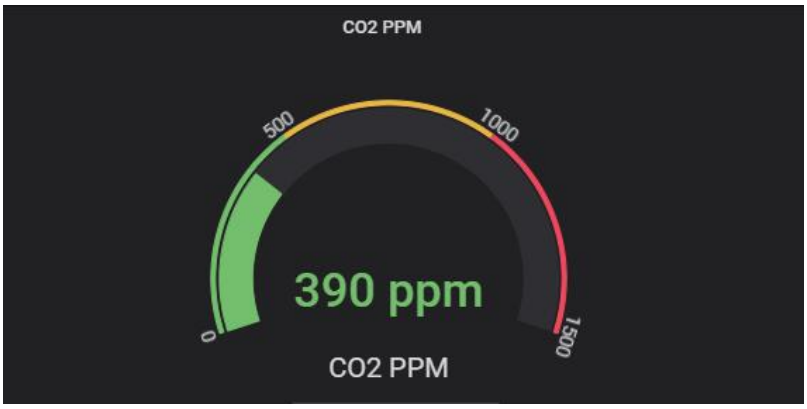

Figura 32: Indicador de concentración de gas CO_2

Cabe destacar que también se muestran otros valores secundarios como máximos, mínimos, medias, etc. Adicionalmente, los paneles son interactivos y permiten visualizar el valor exacto del instante que se desee, así como ampliar o moverse en cualquier intervalo de tiempo almacenado en la base de datos.

Para facilitar la visualización, también se pueden añadir marcadores de eventos, como los que se muestran en azul en la Figura 28. Sirven, por ejemplo, para marcar el instante en que se conecta cierta carga y, de este modo, se pueden revisar los valores a posteriori.

5.5.3.1 Mecanismos de aviso al usuario

Grafana permite establecer alertas para los paneles del *dashboard*, encargándose de hacer peticiones a la base de datos mientras el servicio sigue trabajando en segundo plano.

El primer paso es establecer el canal de comunicación: hay multitud de servicios disponibles: desde correo, Telegram, Slack, Discord, Microsoft Teams, Google Hangouts, etc.

Una vez configurados los canales de comunicación de los usuarios, hay que establecer las reglas que se evaluarán para activar la alerta. Las reglas están compuestas por un nombre, el intervalo de evaluación, la duración de la evaluación y las condiciones.

En el ejemplo de la Figura 33, se crea una alerta que enviará una notificación cuando se detecte un exceso de potencia consumida por encima de 150W. Esta condición se evaluará cada minuto, y si se cumple durante un minuto entero se activará la alerta y el usuario recibirá una notificación.

Figura 33: Ejemplo de configuración de alerta en Grafana

6 Elaboración del prototipo

Para comprobar su correcto funcionamiento, el prototipo se ha armado sobre una placa de pruebas como se muestra en la Figura 34.

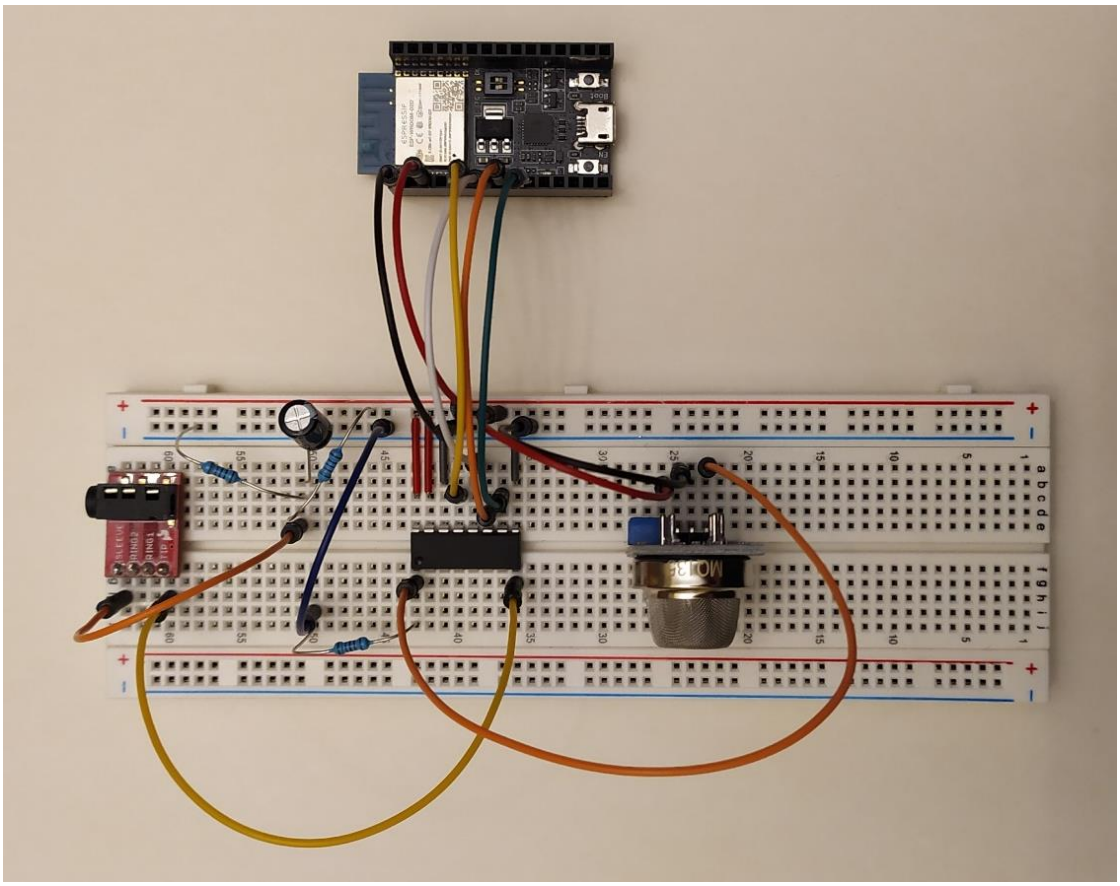

Figura 34: Prototipo montado en una placa de pruebas

Todas las pruebas se han realizado en este montaje temporal dada la facilidad a conectar rápidamente los distintos dispositivos, así como el poder tomar medidas directamente en las conexiones ofrecidas por la placa.

Para comprobar el correcto funcionamiento del ADC se ha visualizado en el osciloscopio sus señales y comprobado que el código utilizado para el protocolo SPI cumple las especificaciones estipuladas en el *datasheet* del ADC. En el Anexo B del documento "Anexos" se muestra dicha comprobación.

6.1 Encapsulado

Una vez comprobado el correcto funcionamiento del prototipo montado en la placa de pruebas, se ha confeccionado un segundo prototipo el cual se alojará en un encapsulado para carril DIN de 35 mm TS-35 (Figura 35). Esto permite su colocación final en un cuadro eléctrico como el de la Figura 36. Así se consigue una integración con el resto de sistemas eléctricos de la instalación, haciendo un uso del espacio mas eficiente y manteniendo un único punto de control eléctrico.

Figura 35: Encapsulado del dispositivo

El encapsulado compatible con el carril DIN de 35 mm tiene unas dimensiones de 90,2 mm de ancho, 53,3 mm de largo y 57,5 mm de alto. Las dimensiones detalladas de este, así como de la placa base que se introducirá en su interior (Figura 37) con los componentes están incluidos en el Anexo G.

Figura 36: Cuadro eléctrico con carril DIN y cables visibles

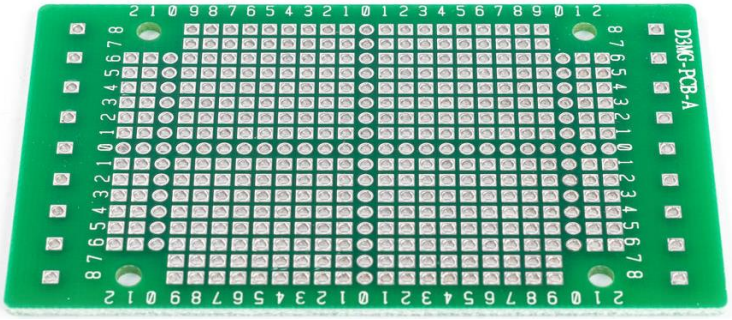

Figura 37: Placa base del encapsulado

Para el funcionamiento del encapsulado y con el propósito de dar autonomía al dispositivo respecto a una fuente de alimentación exterior, se ha añadido una fuente de alimentación AC-DC (TSP-03) que permite convertir los 230 V de corriente alterna de la red eléctrica a los 3,3 V en corriente continua requeridos para la operación del microcontrolador. Así, se alimenta el dispositivo mediante una toma de corriente normal (Figura 38), sin requerir una fuente externa de alimentación, tal como la fuente de banco o la conexión mediante USB a un ordenador usadas en el primer montaje.

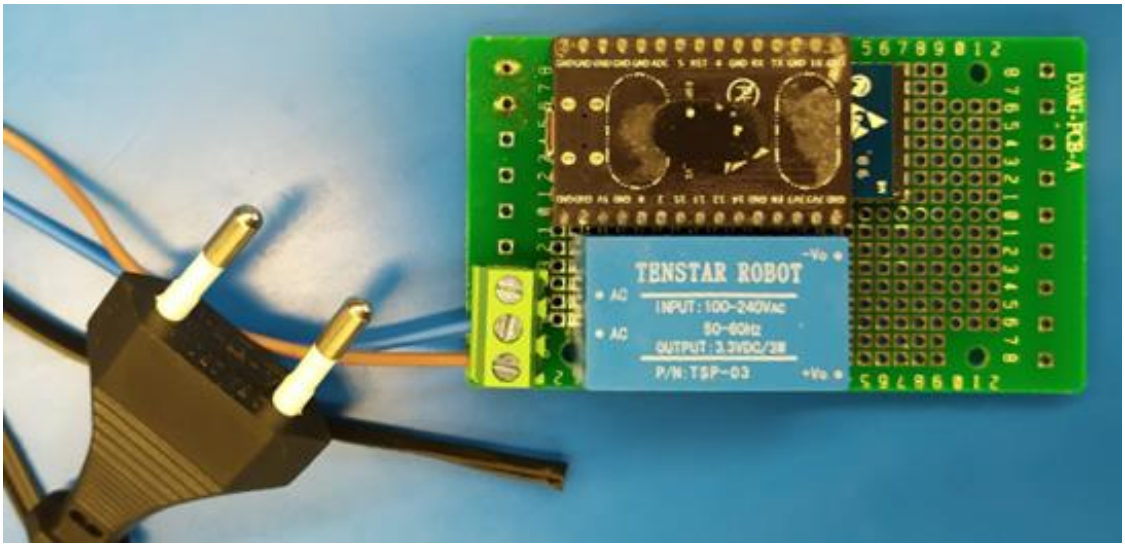

Figura 38: Montaje preliminar usando la placa base y la fuente de alimentación

El encapsulado se puede montar en el cuadro eléctrico, en un espacio disponible en el carril DIN. Así, la colocación del sensor de corriente es directa al conductor que se desee monitorizar, ya que su acceso es desde dentro del mismo cuadro, y la toma de corriente se puede ocultar fácilmente usando un circuito disponible.

Finalmente, después de soldar los componentes y comprobar el funcionamiento del dispositivo, se muestra el dispositivo preparado para su colocación en el cuadro eléctrico en la Figura 39.

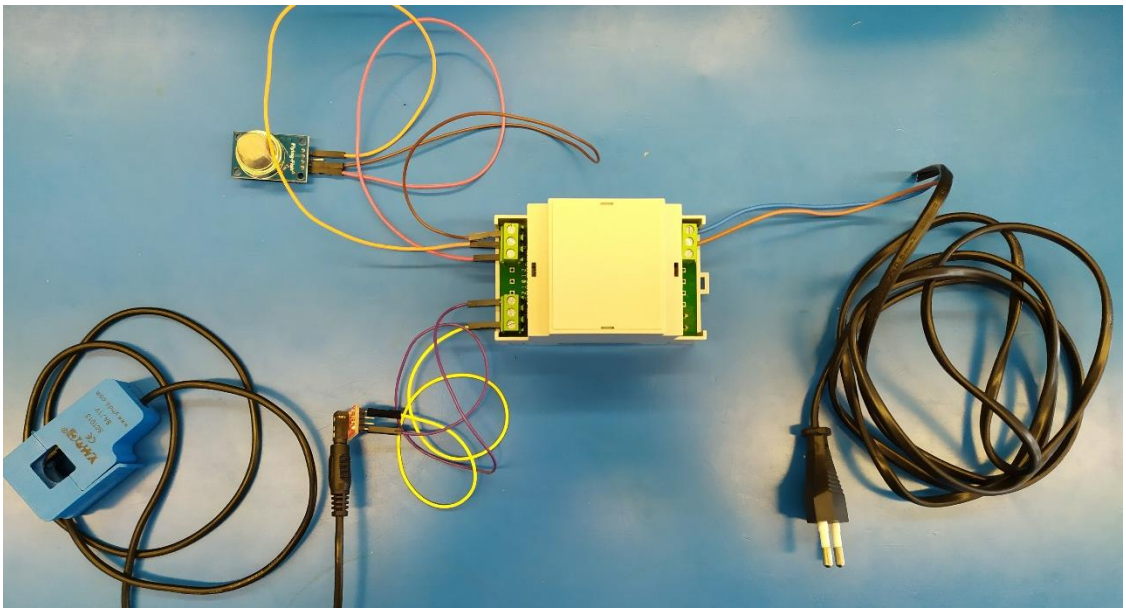

Figura 39: Dispositivo encapsulado con los sensores conectados

6.2 Conexiones físicas

El sensor SCT-013 dispone de un cable de un metro con una terminación en TRS (Punta-Anillo-Cuerpo, del inglés *Tip-Ring-Sleeve*) de 3,5 mm. Para conectarlo al ADC se requiere o bien cortar el cable para separarlo en sus dos terminales o usar una placa adaptadora o *Jack breakout board* que expone los terminales. Se ha escogido esta última opción ya que permite modificar más fácilmente el circuito, así como conectar y desconectar fácilmente el sensor.

La punta del conector TRS que lleva la señal se lleva a un canal del ADC mientras que el cuerpo se conecta al circuito de desplazamiento de voltaje por los motivos descritos en el capítulo anterior.

Figura 40: Conector TRS y placa adaptadora TRRS utilizada

Se ha decidido conectar el sensor de gas al ADC usando el terminal analógico, requiriendo a parte una conexión a tierra y otra a la salida de tensión del ESP8266.

La conexión del ADC es directa al microcontrolador al no haber ningún otro dispositivo conectado mediante SPI. Solo hace falta conectarlo con los 4 pines SPI más la conexión a tierra y tensión del microcontrolador.

Todas las conexiones se muestran en la Figura 41.

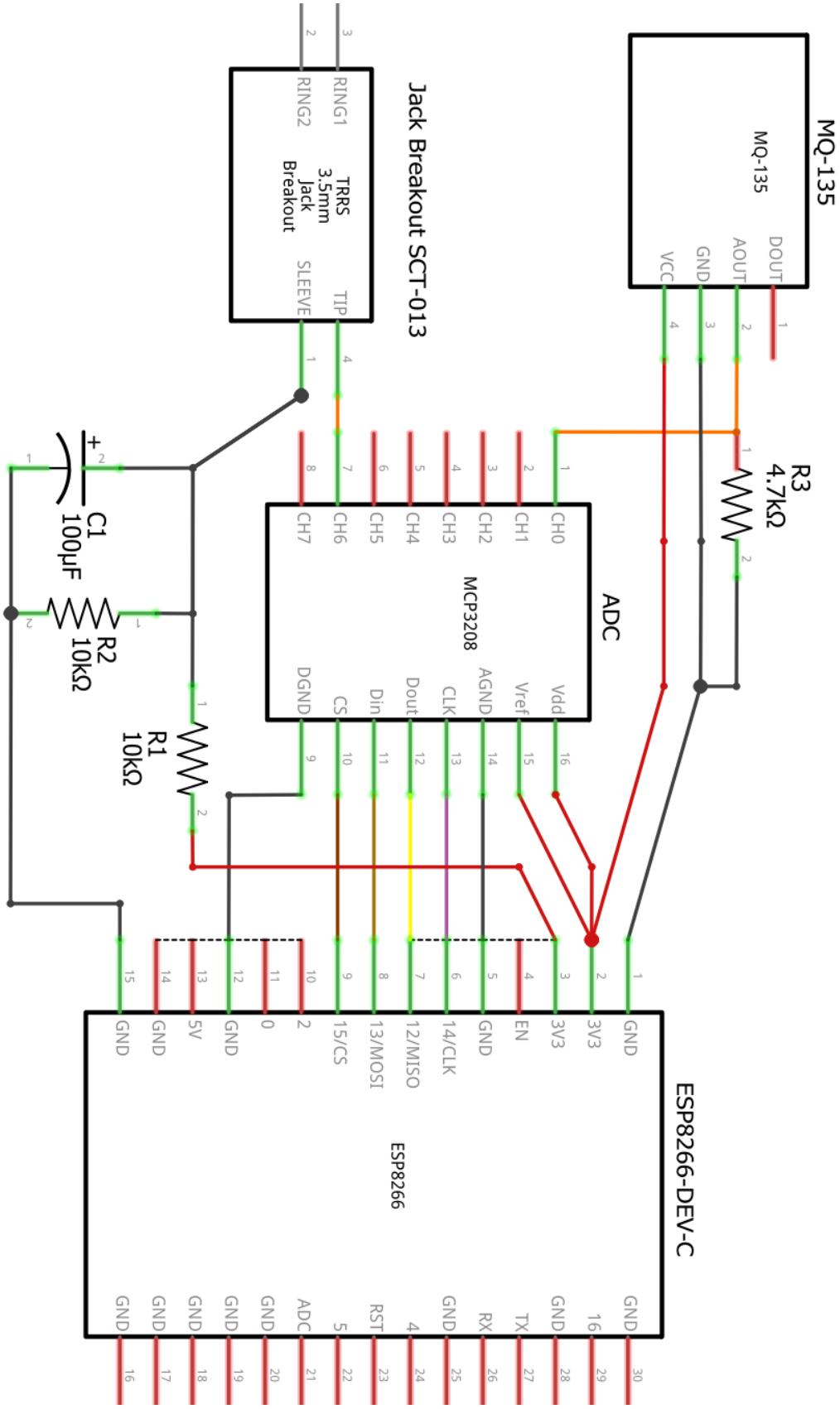

Figura 41: Esquema eléctrico de las conexiones entre dispositivos

6.3 Coste del prototipo

El coste del prototipo se desglosa en la Tabla 10. El coste del microcontrolador utilizado es superior al del producto final ya que es una versión de desarrollo que incluye los circuitos de alimentación, así como pines hembra soldados para facilitar el prototipado. Esto supone un incremento de unos 5 € respecto los microcontroladores ESP8266 utilizables en la versión final.

Producto	Función	Precio
ESP8266 DEV-C	Microcontrolador	7,20 €
Microchip MCP3208	ADC	3,10 €
SCT-013	Sensor de corriente	4,20 €
MQ-135	Sensor de gas CO2	3,40 €
Conectores tornillo a placa base (x3)	Conexiones SCT-013, MQ-135 y fuente de alimentación	0,36 €
TSP-03 230V -3,3V	Fuente de alimentación AC-DC	1,80 €
D3MG	Encapsulado Carril DIN	2,75 €
D3MG-PCB-A	Placa base encapsulado	2,80 €
Cable USB	Conexión microcontrolador	Reutilizado
Electrónica varia	Circuito electrónico, resistencias, condensadores, cables, soldadura	Reutilizado
Total		26,61 €

Tabla 10: Desglose del coste del prototipo final encapsulado

7 Pruebas y resultados

A continuación, se presentan las pruebas realizadas en cuanto al funcionamiento del dispositivo, así como la verificación de los resultados obtenidos.

7.1 Programación a prueba de errores

La programación del microcontrolador contempla varios mecanismos para asegurar la correcta transmisión de los datos, manteniendo el sistema de comunicaciones a prueba de fallos. En el caso de que se genere algún problema, el microcontrolador intentará solucionarlo, así como hacer posible un diagnóstico de los fallos más sencillo a través de códigos de error y mensajes por el puerto serial.

En el caso que el microcontrolador se quede sin conexión WiFi, este responderá adecuadamente, enviando un mensaje por el puerto serial e intentando conectarse otra vez. Esto se consigue mediante la condición en el bucle principal de comprobar la conexión e intentar restablecerla cuando no se esté conectado a ninguna red. Antes de volver a intentar la conexión se aplica un retraso mediante la función *delay().* De este modo, se intenta evitar que el microcontrolador pueda quedarse atascado.

En la Figura 42 se muestra el diagrama de flujo del programa con especial énfasis en los mecanismos programados que otorgan robustez al sistema de comunicaciones. Como se explicó en el capítulo Programación del microcontrolador, la función *yield*() se usa para ceder el paso a procesos en segundo plano, lo cual es vital para evitar que se resetee el microcontrolador debido a los WDT, cuyo tiempo de espera por defecto es de unos 2 segundos. Dependiendo de la cantidad de datos que se quiera guardar en la base de datos a lo largo del tiempo, se le puede dar el valor deseado al retraso llamado mediante la función *delay*(). El TCP *timeout* por defecto es de 5 segundos, que es el tiempo que se espera antes de imprimir el error si no ha recibido respuesta HTTP de la base de datos.

A continuación, se explican las pruebas realizadas para contrastar el correcto funcionamiento de los mecanismos contemplados.

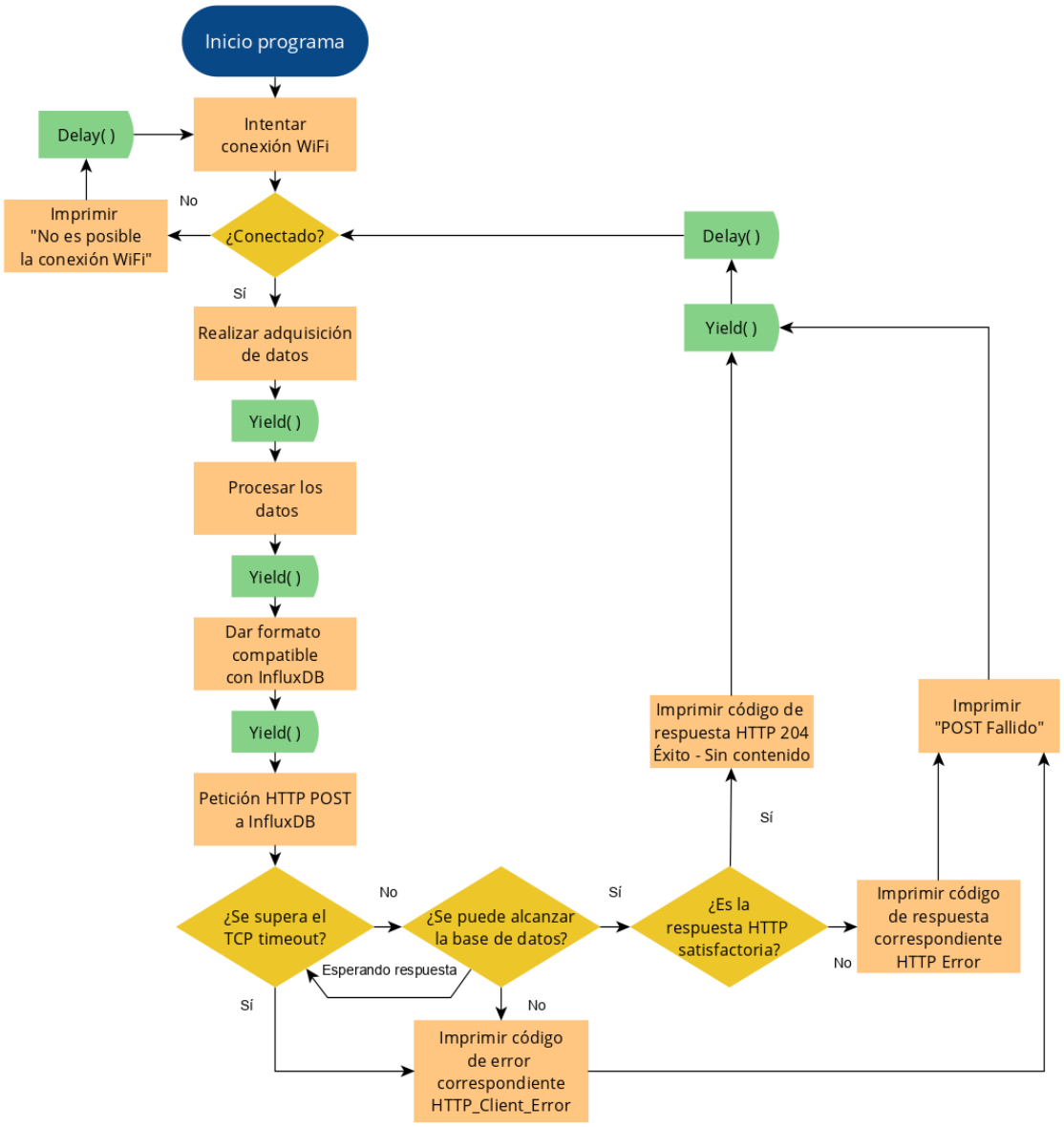

Figura 42: Flujograma del programa con detalles de la programación a prueba de errores

En la Figura 43 se puede observar un ejemplo de la respuesta del microcontrolador a través del puerto serial en el momento en el que se fuerza la desconexión a la red WiFi, así como la posterior recuperación de la conexión, cuando el sistema vuelve a funcionar correctamente. Este mecanismo corresponde a la primera toma de decisión del programa, como se ilustra en el flujograma de la Figura 42.

```
 <-- Response: 204 ""
 --> writing to test:
 <-- Response: 204 ""
 --> writing to test:
 <-- Response: 204 ""
 --> writing to test:
 <-- Response: 204 ""
No es posible la conexion WiFi
No es posible la conexion WiFi
No es posible la conexion WiFi
No es posible la conexion WiFi
No es posible la conexion WiFi
 --> writing to test:
 <-- Response: 204 ""
 --> writing to test:
 <-- Response: 204 ""
```

Figura 43: Ejemplo de simulacro de pérdida de conexión WiFi

De un modo parecido, el programa da un error a través del puerto serial si la información no se ha podido enviar correctamente a InfluxDB, devolviendo el código de respuesta HTTP que le llega desde InfluxDB. El código se puede utilizar para encontrar el error, o confirmar que el funcionamiento es el correcto en caso que el código de respuesta sea el 204 (Todo correcto, la respuesta no tiene cuerpo). Este mecanismo corresponde a la última toma de decisión del programa tal como se indica en el flujograma de la Figura 42.

En el caso de que el problema no se encuentre en el protocolo HTTP, por ejemplo, si la base de datos no está operativa, el microcontrolador seguirá intentando la conexión hasta que se pueda conectar, dando el código de error correspondiente de la Tabla 11 a través del puerto serial. Este mecanismo corresponde a la segunda toma de decisión del flujograma de la Figura 42.

En la Figura 44 se muestra el caso en el que se cierra el servicio InfluxDB, y la conexión es denegada debido a que no hay ninguna aplicación escuchando en el puerto de acceso del servidor especificado. En consecuencia, se recibe el código de error correspondiente hasta que se consigue reanudar la comunicación.

Error	Código de error
Conexión denegada	-1
Error de envío de la cabecera	-2
Error de envío del cuerpo	-3
No conectado	-4
Conexión perdida	-5
Error de trama de datos	-6
Sin servidor HTTP	-7
RAM insuficiente	-8
Error de codificado	-9
Error de escritura	-10
Expiración tiempo de espera TCP	-11

Tabla 11: Códigos de error de la librería InfluxDB (Schuerg, 2019)

```
 --> writing to test:
 <-- Response: 204 ""
 --> writing to test:
 <-- Response: -1 ""
#####
POST FAILED
#####
 --> writing to test:
 <-- Response: -1 ""
#####
POST FAILED
#####
 --> writing to test:
 <-- Response: -1 ""
#####
POST FAILED
#####
 --> writing to test:
 <-- Response: 204 ""
 --> writing to test:
 <-- Response: 204 ""
```

Figura 44: Ejemplo del error recibido al cerrar el servicio InfluxDB

En la figura Figura 45 se muestra un ejemplo donde se recibe el código de error HTTP correspondiente a un error de autentificación, provocado al introducir las credenciales de acceso a la base de datos incorrectas.

```
Connecting to WIFI..........WiFi conectado, SSID:
MOVISTAR_4052
IP ESP8266:
192.168.1.41
Setup done
 --> writing to test:
 <-- Response: 401 "{"error":"authorization failed"}
"
#####
POST FAILED
#####
 --> writing to test:
 <-- Response: 401 "{"error":"authorization failed"}
"
#####
POST FAILED
#####
```

Figura 45: Ejemplo de autorización denegada y código HTTP correspondiente

Con estos mecanismos, se obtiene un programa robusto que intenta recuperar el correcto funcionamiento del sistema de monitorización.

7.2 Validación de los resultados obtenidos

Los datos obtenidos por los sensores se encuentran dentro de lo esperado. En el caso del sensor SCT-013, se han realizado pruebas para validar los consumos obtenidos conectando varios dispositivos a la red monitorizada.

El sensor SCT-013, según su *datasheet* (Anexo C), sólo garantiza una no-linealidad de menos del 3% en un rango de medidas desde el 10% al 120% de su entrada de corriente nominal. En este caso se está trabajando con el SCT-013-005, que tiene una entrada de corriente nominal de hasta 5 A, por lo que sólo se obtendrán resultados correctos a partir de unos 0,5 A, como se ha podido constatar durante las pruebas.

Para validar el consumo solventando el problema de la no-linealidad en la parte baja del rango de adquisición, se ha medido la corriente eléctrica consumida por un secador de pelo a media potencia para elevar el consumo por encima de los 0,5 A, junto con un tubo fluorescente de 11 W que se ha desconectado posteriormente.

La diferencia entre la corriente consumida antes y después de la desconexión del fluorescente es de $2{,}762\ A\ -\ 2{,}710\ A = 0{,}052\ A$, como se muestra en la Figura 46.

Si se toma como aproximación que la tensión en la instalación es de 230 V, se obtiene que el consumo de la lámpara (Figura 47) es de $0{,}052\ A * 230\ V = 11{,}96$ W, lo cual cuadra con el consumo esperado, ya que no se conoce el consumo exacto de la lámpara.

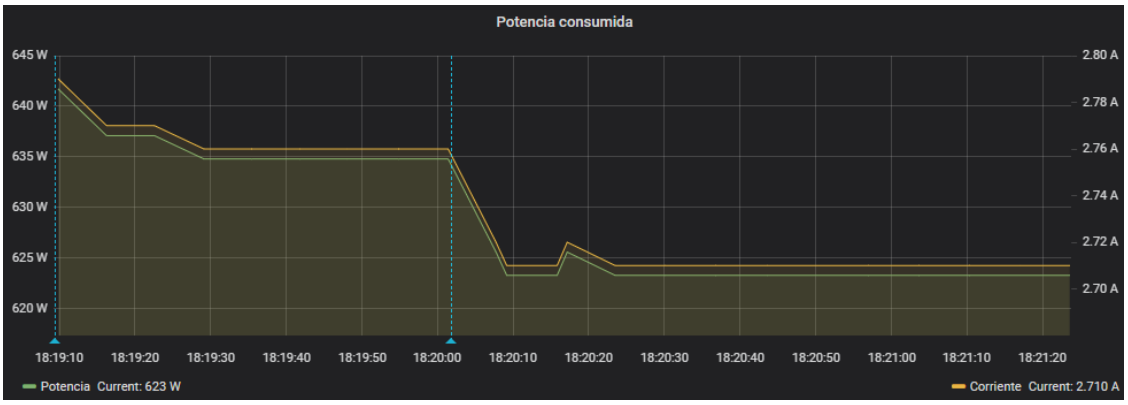

Figura 46: Potencia y Corriente antes y después de la desconexión de una lámpara de 11 W

Figura 47: Características de la lámpara utilizada

Por otro lado, los datos obtenidos tras el análisis de frecuencia mediante la FFT también entran dentro de lo esperado, se detecta correctamente la frecuencia de la red eléctrica de 50 Hz en su frecuencia fundamental, siendo la magnitud de los armónicos mucho menor que esta, como es de esperar. Dependiendo de la carga que conectemos, la amplitud de los armónicos será diferente. Igualmente, el análisis de frecuencia es capaz de distinguir estos armónicos, ya que se pueden ver claramente.

En la Figura 48, al conectar una carga no lineal (ordenador portátil) se puede apreciar claramente el aumento de consumo en la frecuencia fundamental (50 Hz),

así como una segunda banda de frecuencias en el tercer armónico (150 Hz) y otra más leve en el sexto armónico (300 Hz).

En la Figura 49 se aprecia la aparición del segundo, cuarto y quinto armónico al conectar un cargador de móvil de 10 W, junto con un leve aumento de la magnitud del primer y tercer armónicos ya presentes antes de la conexión del dispositivo. Esto también se ve claramente representado en el gráfico adjunto en la misma figura que muestra la THD, su aumento y disminución al conectar y desconectar respectivamente el dispositivo.

Figura 48: Aumento de consumo y aparición de harmónicos al conectar a la red monitorizada un ordenador portátil como carga no lineal

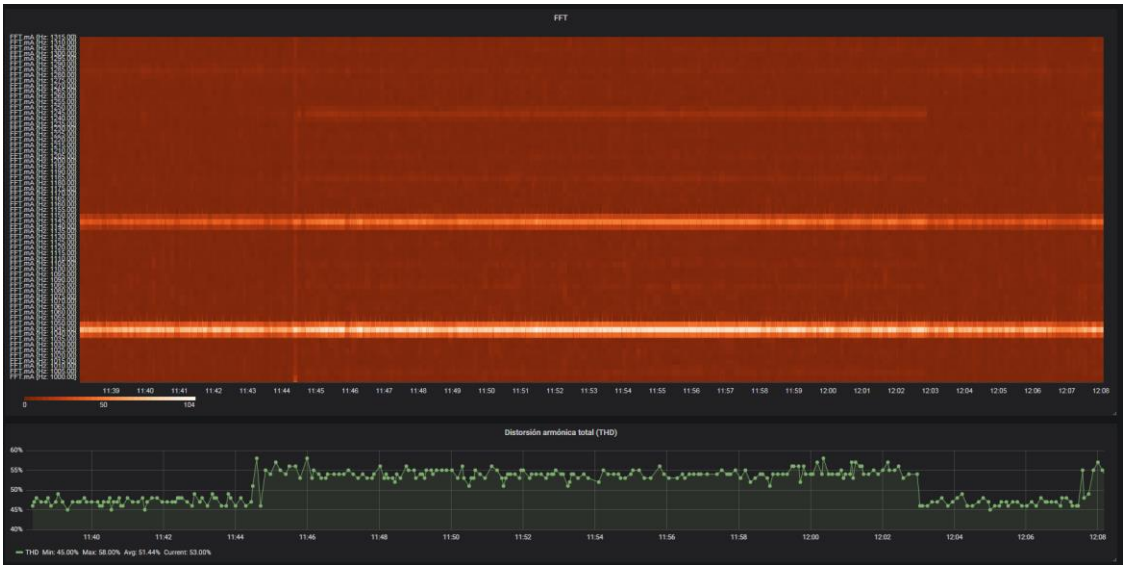

Figura 49: Efecto sobre la distorsión armónica de conectar un cargador de móvil al circuito monitorizado y desconectarlo. Se aprecia el aumento de la THD así como los nuevos armónicos que aparecen

La FFT identifica correctamente la frecuencia fundamental de la red monitorizada, situada en torno a los 50.4 Hz (Figura 50), lo cual entra dentro de los márgenes posibles de frecuencia, que debe estar alrededor de 50 Hz.

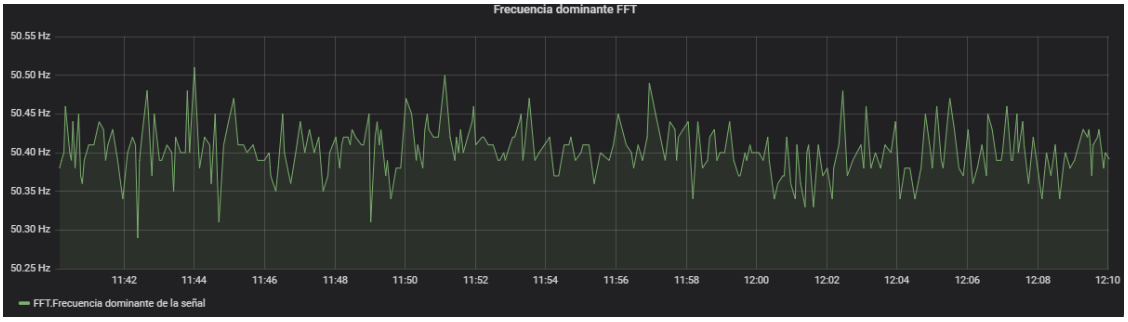

Figura 50: Frecuencia dominante en el análisis de frecuencias de la FFT

El sensor de gas muestra el comportamiento esperado después de la calibración con los datos del *datasheet* y una vez pasado un período de calentamiento del sensor. Para calentarse, el sensor dispone de una resistencia de 800 mW según su *datasheet* (mostrada en el anexo D) que debe estar alimentada constantemente de modo que el sensor esté dentro de su rango de temperatura operativa para el cual está calibrado.

7.3 Verificación de los resultados

La verificación de los resultados debe corroborar que las medidas realizadas por los sensores sean precisas, comparando, para ello, con un equipo calibrado con anterioridad. La verificación se debe llevar a cabo para cada medida realizada: las del ámbito del consumo de corriente, los resultados de la FFT y para el sensor de concentración de gas CO_2.

En primer lugar, el procedimiento para verificar los resultados del consumo de corriente consiste en monitorizar con el prototipo y a la vez con un multímetro la corriente consumida. El multímetro debe ir colocado en serie con el cable de fase de la red. Comparando las corrientes leídas por el sensor y por el multímetro se puede calibrar el prototipo y verificar su correcta lectura.

En segundo lugar, para verificar la magnitud de las frecuencias obtenidas a con la FFT se requiere el uso de un analizador de espectro junto con una sonda de corriente. De este modo se puede comprobar si el análisis frecuencial realizado es correcto, la frecuencia fundamental corresponde con la medida, los armónicos que aparecen son de la magnitud correcta y la THD coincide con el valor real.

Finalmente, para verificar el sensor de CO_2 se requiere conocer la concentración de este gas en varios puntos, bien usando un sensor calibrado más preciso o bien conociendo la concentración de varios ambientes.

Estas verificaciones deben realizarse por cada sensor utilizado, por ejemplo, en el prototipo se ha estado trabajando con el sensor SCT-013 de 5 A, pero el diseño final se beneficia más de un SCT-013 de más amperaje, por lo que las pruebas deberían realizarse con el nuevo sensor ya que sus características son diferentes.

7.4 Cuestiones de seguridad

El presente trabajo ha requerido la manipulación de cables conectados a la red eléctrica. Gracias al uso de un sensor de corriente no invasivo, no se ha requerido cortar ningún cable y tan solo ha sido necesario exponer los cables internos, cortando el recubrimiento protector de plástico exterior del cable como se muestra en la Figura 51.

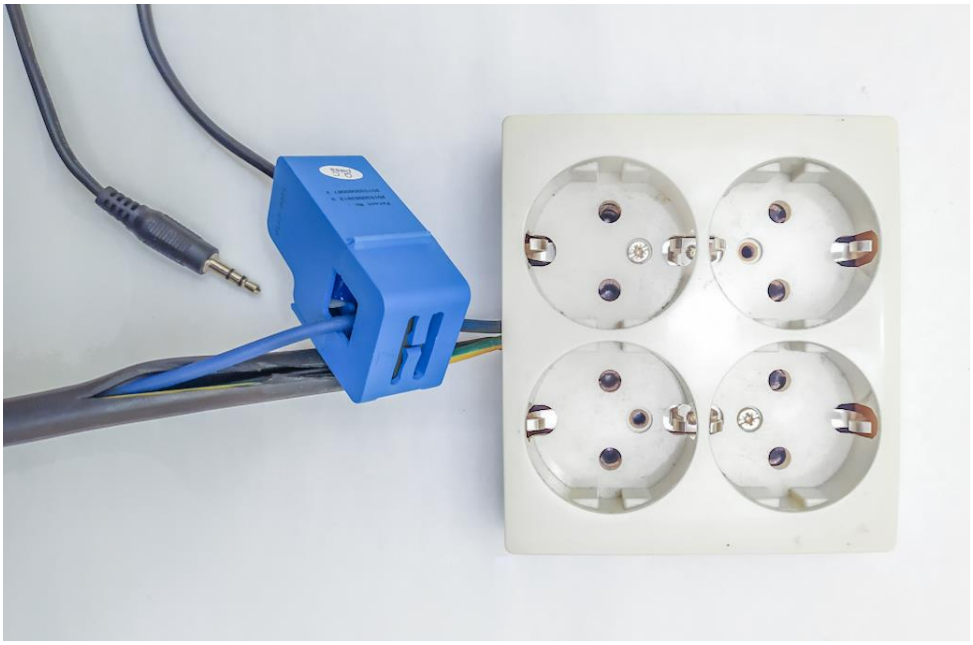

Figura 51: Manipulación de los cables para la colocación del sensor de corriente SCT-013

Para realizar la verificación de la medida del consumo de corriente eléctrica sí sería necesario exponer los hilos conductores de cobre ya que se requiere de una conexión física en serie con el multímetro.

La manipulación de los demás componentes electrónicos no supone un peligro debido a la naturaleza del trabajo, ya que se ha trabajado en electrónica de baja potencia, con voltajes y corrientes bajos. Tan solo se ha requerido el uso correcto de los instrumentos de medición como osciloscopios, por ejemplo, en el caso de comprobación del ADC, y evitar cortocircuitos y sobrecargas al usar fuentes de alimentación externas.

Finalmente, para las soldaduras realizadas también ha sido necesario utilizar el equipamiento de la estación de soldadura correctamente, así como usar el equipo de protección individual (EPI) necesario, como guantes, bata, extracción de gases y protección ocular.

8 Impacto medioambiental

El estudio realizado, así como su implementación, construcción del prototipo y pruebas no han consumido una gran cantidad de recursos energéticos. El mayor impacto se ha debido al consumo eléctrico del portátil con el que se ha trabajado. La potencia media consumida durante el uso que se le ha dado al portátil, el cual no incluye grandes esfuerzos de computación, es de entre 25-55W. La mayor parte del trabajo ha requerido el uso del ordenador, siendo prácticamente las 611 horas del trabajo con este encendido.

En cuanto al transporte necesario, se han realizado unos 15 viajes en tren. Según los datos proporcionados por los *Ferrocarrils de la Generalitat de Catalunya*, sus trenes garantizan un consumo 100% eléctrico, con un consumo medio de 0,024 kWh por km y usuario, lo que resulta en un consumo de 0,76 kWh por viaje de ida o 1,52 kWh para ida y vuelta.

El equipamiento técnico, así como los componentes electrónicos utilizados, han sido proporcionados por el Laboratorio MCIA de la UPC o ya se disponía de ellos.

La energía requerida para la soldadura de los componentes es despreciable ya que el tiempo de soldadura ha sido muy reducido. El consumo debido a la iluminación también es despreciable debido a que fuese en el laboratorio o en casa se aprovecharon estancias ya iluminadas.

Finalmente, con estos datos, se ha elaborado la Tabla 12, teniendo en cuenta que cada kWh consumido equivale a la emisión a la atmosfera de unos 321 gramos de CO_2 por kWh consumido, según el mix de la red eléctrica española en 2018. (GENCAT, 2018)

Concepto	**Unidades**	**Energía/Ud**	**Energía**	CO_2
Ordenador	611 h	40 [W]	24,44 [kWh]	7,85 kg
Transporte	15 viajes	1,52 [kWh]	22,8 [kWh]	7,32 kg
Total	-	-	47,24 [kWh]	15,2 kg

Tabla 12: Resumen de los elementos con impacto medioambiental

Se puede observar en la tabla que más o menos tanto el transporte como el ordenador han tenido el mismo impacto. La energía consumida no ha sido muy elevada, como es natural en un estudio de estas características.

El consumo del microcontrolador se puede despreciar debido a que es muy bajo y no ha estado en funcionamiento una cantidad de tiempo significativa, tan solo para hacer pruebas de código y funcionamiento.

Como comentario adicional, aunque se ha diseñado este sistema con vistas a aplicaciones medioambientales y de sostenibilidad, no entra dentro del alcance de este estudio estimar la reducción del impacto medioambiental que el sistema pueda conllevar.

9 Resumen del presupuesto

El presupuesto completo se incluye en el documento aparte "Presupuesto", a continuación, se incluye la Tabla 13 a modo de resumen.

Como se puede observar, el coste principal recae en el sueldo, mientras que la magnitud de los demás costes es notablemente inferior.

Concepto	Coste
Coste del estudio	7.560 €
Coste del prototipo	25,61 €
Costes adicionales	107,15 €
TOTAL	7692,76 €

Tabla 13: Coste total del estudio

Futuras líneas de trabajo

Las futuras líneas de trabajo después de este estudio se basan principalmente en añadir funcionalidades adicionales e implementar un post-proceso de los datos más intensivo en pos de lograr una reducción de consumo energético en sistemas HVAC e iluminación.

En primer lugar, sería interesante implementar controles directos a los sistemas HVAC y al sistema de iluminación del edificio, utilizando el sensor de CO_2 como sensor de ocupación de la sala, y con sensores de temperatura adicionales e información del tiempo exterior extraída de APIs de bancos de meteorología.

En segundo lugar, otra línea de trabajo consistiría en administrar la información de los sensores y la información meteorológica exterior con tal de hacer un control inteligente de la energía consumida por el edificio mediante el análisis de tendencias en los datos.

Por último, también cabe mencionar que se podría trabajar en mejorar la seguridad de las comunicaciones, que como se comentó en los capítulos anteriores no está cifrada, dado que no se dispone del certificado SSL requerido que permite utilizar HTTPS.

En cualquier caso, la plataforma de comunicaciones diseñada y desarrollada en el presente trabajo permite encarar problemas de características similares, en los que se requiera recoger datos de sensores inalámbricamente para su monitorización, visualizado y potencial control mediante actuadores que podrían ser incorporados. Estos problemas se encuentran en multitud de campos, no tan solo energéticos sino, por ejemplo, en agricultura, medicina, industria, domótica o investigación.

Conclusiones

El estudio realizado a lo largo de este documento ha permitido el diseño de un sistema de comunicaciones escalable, a prueba de errores y flexible. Este ha sido aplicado con éxito en la monitorización del consumo de energía, calidad de red y monitorización de concentración de CO_2.

Para diseñar este sistema, primeramente se han estudiado en profundidad los modelos teóricos en los que se ha basado el sistema de comunicación, así como los protocolos usados y sus mecanismos que proporcionan fiabilidad.

A partir de esta base teórica se ha justificado la elección de una base de datos (InfluxDB) y una herramienta de visualización (Grafana) adecuadas a la función de monitorización del dispositivo, que proporcionan varias ventajas al sistema, como la optimización de grandes volúmenes de datos, visualización eficiente de los datos a través de internet y un sistema de alertas integrado.

A continuación, se ha llevado a cabo el diseño de la arquitectura. Esta prevé la escalabilidad en cuanto el número de microcontroladores a incorporar a partir de las características de la base de datos y de la herramienta de visualización.

Tras diseñar la arquitectura del sistema se ha escogido un microcontrolador entre las alternativas del mercado actual, decidiéndose por el ESP8266, un microcontrolador de bajo coste con WiFi integrado que ha permitido conectar los datos de los sensores a la base de datos después de su procesado. Además, debido a su reducido tamaño, se ha podido instalar en un encapsulado para conectarlo directamente al cuadro eléctrico.

Por lo que respecta a la programación del microcontrolador, se ha repasado la estructura general del programa, así como el diseño de varios mecanismos que informan del estado del microcontrolador e intentan mantenerlo operativo en todo momento, aunque falle la conexión WiFi o la conexión a la base de datos, lo que proporciona robustez al sistema.

En cuanto a los resultados obtenidos, se ha verificado el correcto funcionamiento de la programación a prueba de errores realizada, así como el buen funcionamiento del prototipo, y se han validado con éxito los resultados obtenidos por los sensores, por lo que se satisfacen las condiciones de éxito.

Finalmente se ha discutido el impacto medioambiental del proyecto, así como las futuras líneas de trabajo realizables.

En definitiva, el sistema de comunicaciones diseñado tiene un bajo coste, lo que hace interesante su uso en situaciones reales. El sistema puede ser ampliado fácilmente, proporciona una visualización eficiente de los parámetros monitorizados y facilita la administración de estos gracias al sistema de alertas al usuario. Además, ha sido probado con éxito, cumpliendo el alcance y los requerimientos establecidos, y tiene un uso potencial en otras disciplinas.

Bibliografía

Bader, A., Kopp, O., & Falkenthal, M. (2017). Survey and comparison of open source time series databases. *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft Fur Informatik (GI)*, *266*, 249–268.

Bottaccioli, L., Aliberti, A., Ugliotti, F., Patti, E., Osello, A., Macii, E., & Acquaviva, A. (2017). *Building Energy Modelling and Monitoring by Integration of IoT Devices and Building Information Models. Proceedings - International Computer Software and Applications Conference, 1, 914–922. https://doi.org/10.1109/COMPSAC.2017.75*

Choi, M. I., Cho, K., Hwang, J. Y., Park, L. W., Jang, K. H., Park, S., & Park, S. (2017). Design and implementation of IoT-based HVAC system for future zero energy building. *2017 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2017*, (20154030200860), 605–610. https://doi.org/10.1109/PERCOMW.2017.7917631

DB-Engines. (n.d.). DB-Engines Ranking - popularity ranking of time Series DBMS. Retrieved November 28, 2019, from https://db-engines.com/en/ranking/time+series+dbms

GENCAT. (2018). Factor de emisión de la energía eléctrica: el mix. Cambio climático. Retrieved December 16, 2019, from https://canviclimatic.gencat.cat/es/actua/factors_demissio_associats_a_lenergia/

InfluxData. (2018). InfluxDB API reference | InfluxData Documentation. Retrieved November 30, 2019, from https://docs.influxdata.com/influxdb/v1.7/tools/api/

Klyne, G., & Newman, C. (2002). Rfc 3339: Date and time on the internet: Timestamps. *The Internet Society, Request for Comments Jul*.

Kodali, R. K., & Mahesh, K. S. (2016). Low cost ambient monitoring using ESP8266. *Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics, IC3I 2016*, 779–782. https://doi.org/10.1109/IC3I.2016.7918788

Lestari, D., Wahyono, I. D., & Fadlika, I. (2018). IoT based Electrical Energy Consumption Monitoring System Prototype: Case study in G4 Building Universitas Negeri Malang. *Proceedings - 2017 International Conference on Sustainable Information Engineering and Technology, SIET 2017*, *2018-Janua*, 342–347. https://doi.org/10.1109/SIET.2017.8304161

Mark Page ; Maria Molina. (2019). The Mobile Economy 2019, 100. Retrieved from http://www.gsmamobileeconomy.com/GSMA Mobile Economy 2013.pdf

National Instruments. (2016). Understanding FFTs and Windowing. *Instrument Fundamentals: Complete Guide*. National instruments.

Postel, J., & others. (1981a). RFC 791: Internet protocol.

Postel, J., & others. (1981b). RFC 793: Transmission control protocol. September.

Rinaldi, S., Flammini, A., Tagliabue, L. C., & Ciribini, A. L. C. (2019). An IoT framework for the assessment of indoor conditions and estimation of occupancy rates: results from a real case study. *Acta Imeko*, *8*(2), 70. https://doi.org/10.21014/acta_imeko.v8i2.647

Ruano, A., Silva, S., Duarte, H., & Ferreira, P. M. (2018). *Wireless sensors and IoT platform for intelligent HVAC control*. *Applied Sciences (Switzerland)* (Vol. 8). https://doi.org/10.3390/app8030370

Saha, S., & Majumdar, A. (2017). Data centre temperature monitoring with ESP8266 based Wireless Sensor Network and cloud based dashboard with real time alert system. *Proceedings of 2nd International Conference on 2017 Devices for Integrated Circuit, DevIC 2017*, 307–310. https://doi.org/10.1109/DEVIC.2017.8073958

Schuerg, T. (2019). ESP8266_Influx_DB. Retrieved October 10, 2019, from https://github.com/tobiasschuerg/ESP8266_Influx_DB

Serra, J., Pubill, D., Antonopoulos, A., & Verikoukis, C. (2014). Smart HVAC control in IoT: Energy consumption minimization with user comfort constraints. *Scientific World Journal*, *2014*(i). https://doi.org/10.1155/2014/161874

Tanenbaum, A. S., & Wetherall, D. (2011). *Computer networks*. Prentice hall.

Wan, Z., Song, Y., & Cao, Z. (2019). Environment dynamic monitoring and remote control of greenhouse with ESP8266 NodeMCU. *Proceedings of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2019*, (Itnec), 377–382. https://doi.org/10.1109/ITNEC.2019.8729519